INSTITUTO FEDERAL DA PARAÍBA
COORDENAÇÃO DO CURSO SUPERIOR DE
BACHARELADO EM ENGENHARIA ELÉTRICA

ALYSSON BATISTA DE SOUZA

Avaliação de desempenho de códigos polares em redes 5G

JOÃO PESSOA
Outubro de 2024

ALYSSON BATISTA DE SOUZA

Avaliação de desempenho de códigos polares em redes 5G

Trabalho de conclusão de curso submetido à Coordenação do Curso Superior de Bacharelado em engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como parte dos requisitos para a obtenção do grau de Engenheiro Eletricista.

Orientadora: Profª. Drª. Suzete Élida Nóbrega Correia

João Pessoa
Outubro de 2024

Dados Internacionais de Catalogação na Publicação (CIP)
Biblioteca Nilo Peçanha do IFPB, *campus* João Pessoa

S729a Souza, Alysson Batista de.

Avaliação de desempenho de códigos polares em redes 5G / Alysson Batista de Souza. - 2024
55 f. : il.

TCC (Graduação – Bacharelado em Engenharia Elétrica) – Instituto Federal de Educação da Paraíba / Coordenação do Curso Superior em Engenharia Elétrica, 2024.
Orientação : Profa. Dra. Suzete Élida Nóbrega Correia.

1. Avaliação de desempenho – códigos polares. 2. Decodificação por cancelamento sucessivo. 3. Codificação de canal. 4. Teoria da informação. I. Título.

CDU 621.391:004.414.2(043)

Bibliotecária responsável: Lucrecia Camilo de Lima – CRB 15/132

ALYSSON BATISTA DE SOUZA

Avaliação de desempenho de códigos polares em redes 5G

Trabalho de conclusão de curso submetido à Coordenação do Curso Superior de Bacharelado em engenharia Elétrica do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, como parte dos requisitos para a obtenção do grau de Engenheiro Eletricista.

Trabalho de conclusão de curso aprovado pela banca examinadora em João Pessoa, 21 de outubro de 2024

gov.br Documento assinado digitalmente
Suzete Elida Nobrega Correia
Data: 22/10/2024 12:55:32-0300
Verifique em https://validar.iti.gov.br

Profª. Drª. Suzete Élida Nóbrega Correia
Orientador - IFPB

gov.br Documento assinado digitalmente
PATRIC LACOUTH DA SILVA
Data: 22/10/2024 15:55:07-0300
Verifique em https://validar.iti.gov.br

Prof. Dr. PATRIC LACOUTH DA SILVA
Membro da banca - IFPB

gov.br Documento assinado digitalmente
MICHEL COURA DIAS
Data: 22/10/2024 13:51:43-0300
Verifique em https://validar.iti.gov.br

Prof. Me. MICHEL COURA DIAS
Membro da banca - IFPB

João Pessoa, 21 de Outubro de 2024

Agradecimentos

Primeiramente, quero expressar minha gratidão aos meus pais e irmãos pelo apoio incondicional e constante encorajamento ao longo dessa longa jornada acadêmica. Sou imensamente grato ao IFPB e aos professores, que generosamente compartilharam seus valiosos conhecimentos. Em especial, meu agradecimento à professora Suzete Élida Nóbrega Correia, que acreditou no meu potencial e no meu trabalho. Sua orientação foi fundamental para que eu percebesse o quanto a engenharia elétrica pode ser fascinante e enriquecedora.

Aproveito para agradecer aos meus amigos Jefferson, Hélio, Henrique, Iramá, Maria Bruna, Pollyana, Roger, Dawson, Cleiston e Lucas Holanda, que estiveram ao meu lado em cada etapa dessa trajetória. Suas palavras de incentivo, companheirismo e momentos de descontração foram essenciais para manter a motivação e enfrentar os desafios. Juntos, criamos memórias que levarei para a vida toda, e sou grato por ter compartilhado essa experiência com pessoas tão especiais.

"Cedo ou tarde você vai aprender, assim como eu aprendi,
que existe uma diferença entre conhecer o caminho e trilhar o caminho.
(Morpheus)

Resumo

Este trabalho tem como objetivo analisar os códigos polares sistemáticos e sua decodificação por cancelamento sucessivo. Para isso, são investigadas as Taxas de Erro de Bit (BER) de palavras-código moduladas com BPSK e Taxas de Erro de Quadro (FER) transmitidas através de um canal AWGN. Foram avaliados diversos comprimentos de palavras-código, sendo que o melhor desempenho foi alcançado com o tamanho de 2048 bits, considerando uma BER de 10^{-6} e uma FER de 10^{-5}.

Palavras-chaves: Códigos Polares, Decodificação por Cancelamento Sucessivo, codificação de canal.

Abstract

This work aims to analyze systematic polar codes and their decoding by successive cancellation. For this purpose, Bit Error Rates (BER) of codewords modulated with BPSK and Frame Error Rates (FER) transmitted through an AWGN channel are investigated. Several codeword lengths were evaluated, and the best performance was achieved with a length of 2048 bits, considering a BER of 10^{-6} and a FER of 10^{-5}.

Keywords: Polar Codes, Successive Cancellation Decoding, Channel Coding

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

NR	New Radio
IOT	Internet of Things
LDPC	Model, View, Controller
XOR	Exclusive OR
RM	Reed-Muller
LLR	Log-Likelihood Ratios
BPSK	Binary Phase Shift Keying
AWGN	Additive White Gaussian Noise
SC	Successive Cancellation

Sumário

1 Introdução

Os dias atuais correspondem a era da implementação do 5G, a Quinta Geração de Redes Móveis, tal geração foi impulsionada porque as redes atuais não tem estrutura suficiente para suportar o grande tráfego de rede esperado e serviços que estão por vir. A nova geração não se limita a um aumento de velocidade, se difere por abordar também outros parâmetros como latência, eficiência energética e volume de dados que o padrão antecessor não consegue suportar, ressaltando a necessidade de redesenho da arquitetura de rede da operadora. O 5G trará dentro da própria tecnologia uma maior eficiência de transporte de informação na medida terá inteligência para otimizar caminhos de tráfego enquanto escolhe o caminho mediante parâmetros específicos de rede (FARIAS, 2021).

O 5G NR representa um avanço nas capacidades das redes móveis. Até agora, as redes móveis forneceram principalmente conectividade para smartphones, tablets e laptops para consumidores. O 5G levará a banda larga móvel tradicional ao extremo em termos de taxas de dados, capacidade e disponibilidade. Além disso, o 5G permitirá novos serviços, incluindo conectividade para a Internet das Coisas (IoT) industrial e comunicações críticas. As metas do 5G estabelecidas são muito altas, com taxas de dados de até 20 GB/s e aumentos de capacidade de até 1000 vezes, com plataformas flexíveis para a conectividade de dispositivos, latência ultrabaixa e alta confiabilidade. Várias novas aplicações e casos de uso podem ser executados nas redes móveis 5G (HOLMA, 2020).

Os melhores caminhos para o tráfego de bits podem ser obtidos através da polarização de canal de sistemas de comunicações. A polarização de canal é realizada através de códigos polares, que podem entregar até 99,999% de confiabilidade, sendo assim passíveis de serem utilizados para as aplicações que necessitam uma taxa de erro ínfima. Esses códigos podem atingir altas taxas de transferência e baixas taxas de erro, no campo das comunicações digitais, a probabilidade de erros é normalmente expressa por meio do BER (Bit Error Rate), a taxa de bits que foram recebidos com erros em relação à quantidade total de bits que foram transmitidos, ou por meio da FER (Frame Error Rate), a taxa de quadros que foram recebidos com erros em relação à quantidade total de quadros que foram transmitidos, com todas essas características os códigos polares são atrativos para muitos cenários no 5G (MARTãO, 2018).

O Código Polar foi proposto por Erdal Arikan, introduzindo o conceito de polarização de canal. Ele possui como principais características a complexidade linear, na codificação e decodificação, e pode alcançar a capacidade em canais discretos sem memória a partir do aumento do comprimento do código. Assim, é um dos códigos de alto desempenho mais visados, disputando com os códigos Turbo e LDPC (MATTOS, 2018).

Este trabalho propõe a uma avaliação de códigos polares, de forma didática, por meio de uma implementação de uma biblioteca python. Ao longo do projeto, serão abordados os processos de codificação e decodificação de canal, além da avaliação da confiabilidade do sistema por meio do parâmetro BER (Taxa de Erro de Bit) e FER (Taxa de Erro de Quadro). O trabalho inicia com a apresentação dos conceitos fundamentais, como a teoria da informação de Claude Shannon, e os códigos polares e polarização de canal propostos por Arikan, além de métodos de codificação e decodificação, com ênfase em suas características e desempenho.

2 Fundamentação Teórica

Nesta seção, serão relembrados e apresentados os principais conceitos relacionados à Teoria da Informação, Polarização de Canal, Codificação de Códigos Polares e Decodificação de Códigos polares. As referências sobre os conceitos apresentados nesse tópico podem ser encontradas em:(MARTãO, 2018),(LEONARDO., 2019), (THOMAS, 2006), (S.HAYKIN., 2001) e (LIN, 2001), (AL., 2014), (AL., 2019), (TACIANA, 2012),.

2.1 Teoria da informação

Proposta por Claude Shannon em 1948, a Teoria da Informação responde a duas questões fundamentais na teoria da comunicação: Qual é a compressão de dados ideal, ou seja, a quantidade de informação gerada por uma fonte presente no sistema (a entropia H), e qual é a taxa de transmissão ideal de um sistema de comunicação (a capacidade do canal C). Por essa razão, alguns consideram a Teoria da Informação uma subárea da teoria da comunicação.

De início, é necessário conceituar algumas medidas presentes no contexto da Teoria da Informação. As quatro principais são: a própria informação, a entropia, a informação mútua e a capacidade de canal.

2.1.1 Informação:

Diferente de como é usualmente conhecida, na teoria de Shannon, a informação é considerada uma medida de incerteza, estando diretamente relacionada à quantidade de surpresa que um evento possui. Quanto mais surpreendente for um evento, maior será sua quantidade de informação. Por exemplo, a afirmação "nevará no Nordeste brasileiro"possui muito mais informação do que a afirmação "nevará no Sul do Brasil", justamente porque, quanto mais inesperado é um evento, maior é o seu grau de informação.

Matematicamente, a quantidade de informação gerada está relacionada à probabilidade da ocorrência de um evento, sendo definida como:

$$I(x) = log_2\left(\frac{1}{p(x)}\right) \tag{2.1}$$

ou

$$I(x) = -log_2\left(\frac{1}{p(x)}\right) \tag{2.2}$$

onde:

- I(x) = quantidade de informação do evento x
- p(x) = probabilidade de ocorrência do evento x
- log_2 = logaritmo na base 2

Deste modo, a média da quantidade de informação, definida como a medida de informação média por símbolo, pode ser dada da seguinte forma:

$$E[I(x)] = \sum_{i=1}^{M} p(x)I(x) \tag{2.3}$$

Onde a partir da equação 2.4, obtemos a entropia.

2.1.2 Entropia:

A entropia representa o grau de desordem ou incerteza de uma informação, quantificando a quantidade de informação contida em uma mensagem. Quanto maior a informação, maior a desordem e, consequentemente, maior a entropia. Por outro lado, quanto menor a informação, menor a desordem e menor a entropia. O que realmente importa é a quantidade média de informação ao longo do tempo, utilizada para dimensionar corretamente os sistemas de comunicação.

Matematicamente, a entropia é expressa como uma medida em bits por símbolo:

$$H[X] = \sum_{i=1}^{M} p(x) log_2 \left(\frac{1}{p(x)} \right) \tag{2.4}$$

ou

$$H[X] = -\sum_{i=1}^{M} p(x) log_2(p(x)) \tag{2.5}$$

Exemplo 1: Em uma urna existem 4 bolas de diferentes cores: a bola azul tem uma probabilidade de P(verde) = 0.25, a bola verde tem uma probabilidade de P(azul) = 0.5, enquanto a bola vermelha e a amarela possuem um P(vermelha)= P(amarela)= 0.125. Qual a entropia da urna?

$$H[X] = -\sum_{i=1}^{4} p(x) log_2(p(x)) = -0.5 * log_2(0.5) - 0.25 * log_2(0.25) - 2 * (0.125 * log_2(0.125))$$

$$H[X] = 1.75$$

Tal resultado indica que, em média, os símbolos da fonte podem ser representados por 1.75 bits.

Exemplo 2: Em uma corrida de 8 cavalos, assuma que a probabilidade de vitória de cada cavalo são as seguintes: $(\frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64}, \frac{1}{64})$. Qual a entropia da corrida?

$$H[X] = -\sum_{i=1}^{4} p(x_i)\log_2(p(x_i)) = -\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right) - \frac{1}{8}\log_2\left(\frac{1}{8}\right) - \frac{1}{16}\log_2\left(\frac{1}{16}\right) - 4\cdot\left(\frac{1}{64}\log_2\left(\frac{1}{64}\right)\right)$$

$$H[X] = 2$$

Assim como no exemplo anterior, esse resultado indica que, em média, os símbolos da fonte podem ser representados por 2 bits.

Em sistemas de telecomunicações é comum encontrar fontes binárias, ou seja, que emitem apenas 0 ou 1 cuja entropia, supondo que p(0) = p e p(1) = 1-p, é dada por:

$$H[X] = p * log_2(p) - (1-p) * log_2(1-p) \tag{2.6}$$

Para valores de p=0,5, ou seja, onde ambos valores possuem a mesma probabilidade de ocorrer, H(x) = 1bit, a informação média de observação será 1 bit, como ilustra a Fig. 1.

Notamos que a entropia tende a zero nos extremos, onde não há incerteza.

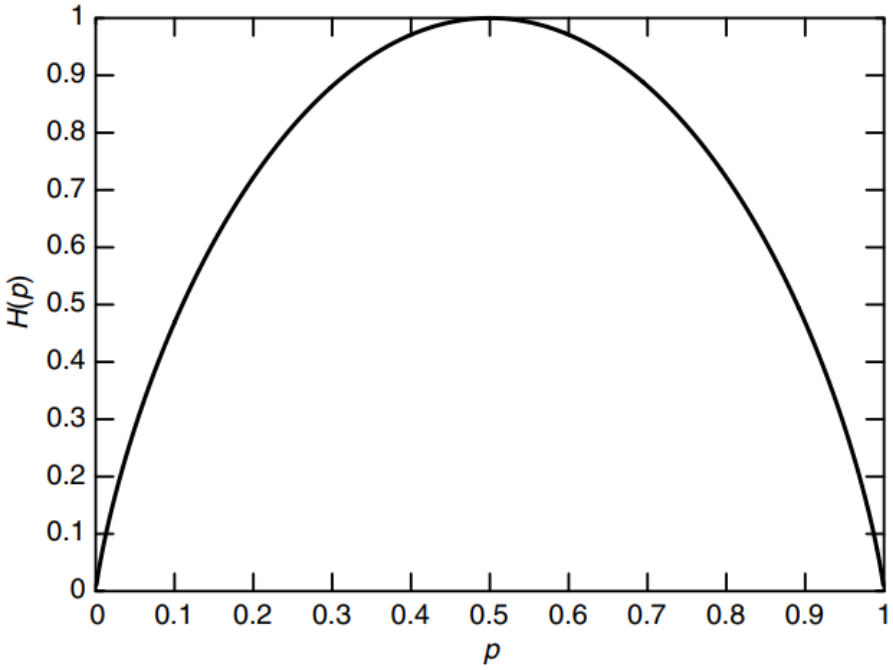
2.1.3 Informação mútua:

Se refere a medida de informação em comum entre duas variáveis aleatórias, essa medida é usada para mensurar o quanto a incerteza associada a uma variável se reduziu, por meio de informações trazidas por outra variável. Todavia, se X for independente de Y, não haverá redução de incerteza, pois não é trazida informação relevante. A informação mútua entre duas variáveis pode ser definida como:

$$I(X;Y) = \sum_{x,y} p(x,y) log_2(\frac{p(x,y)}{p(x) * p(y)}) \tag{2.7}$$

Simplificando a equação 2.7, é obtida uma série de equações que formam um teorema relacionando entropia e informação mútua:

Figura 1 – Gráfico: Entropia X Probabilidade.



Fonte: Elaborado pelo autor

$$I(X;Y) = H(X) - H(X|Y) \tag{2.8}$$

$$I(X;Y) = H(Y) - H(Y|X) \tag{2.9}$$

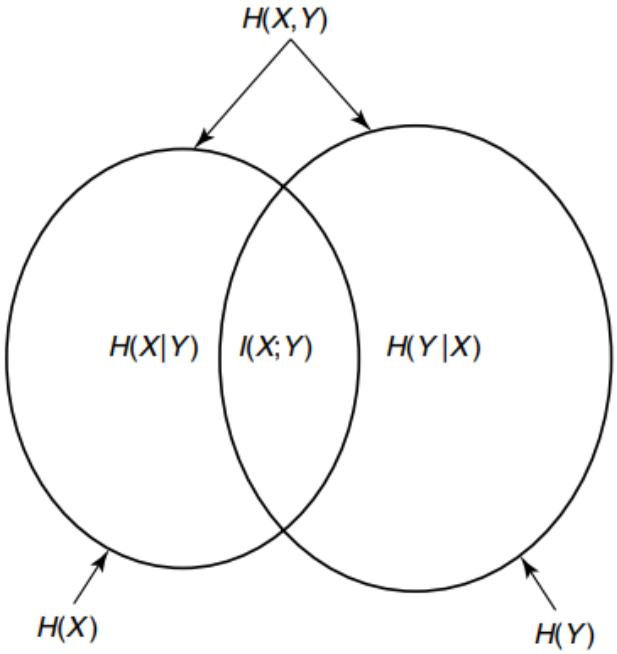
$$I(X;Y) = H(X) + H(Y) - H(X|Y) \tag{2.10}$$

$$I(X;Y) = I(Y;X) \tag{2.11}$$

$$I(X;X) = H(X) \tag{2.12}$$

Na Fig. 2 é demonstrado as relações entre as 5 equações encontradas por meio de um diagram de Venn.

Figura 2 – Diagrama de Venn da Informação Mútua



Fonte: Elaborado pelo autor

A informação mutua é um parâmetro muito importante para a avaliarmos a informação mútua entre a entrada de um canal de comunicação e a sua saída, porque assim é possível definir tal canal como confiável ou não. Se as informações enviadas são dependentes, existirá informação mútua e quanto maior for o grau de informação mútua, maior será a confiabilidade. Caso estas variáveis sejam independentes, haverá redução de incerteza e a informação mútua será nula, caracterizando uma comunicação não confiável.

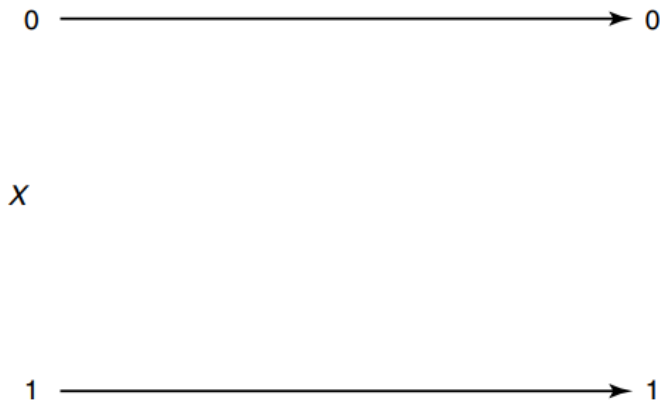
2.1.4 Capacidade de canal:

A capacidade de canal pode ser entendida como espécie de limite de velocidade de transmissão" para estabelecer uma comunicação segura. Caso esse limite seja ultrapassado, pode-se ter erros ou falta de segurança nas informações enviadas. Formalmente é a taxa máxima de informação que este canal suporta, considerando todas as possíveis distribuições das prováveis entradas. Como essa grandeza se vale da informação mútua, a capacidade do canal é medida em bits por uso do canal e pode ser entendida como:

$$C = \max_{\{p(x_j)\}} I(\mathbf{X};\mathbf{Y}) \geq 0$$

Exemplo 3: A Figura 3 apresenta um canal binário sem ruído, caracterizado como um canal ideal.

Figura 3 – Canal Ideal

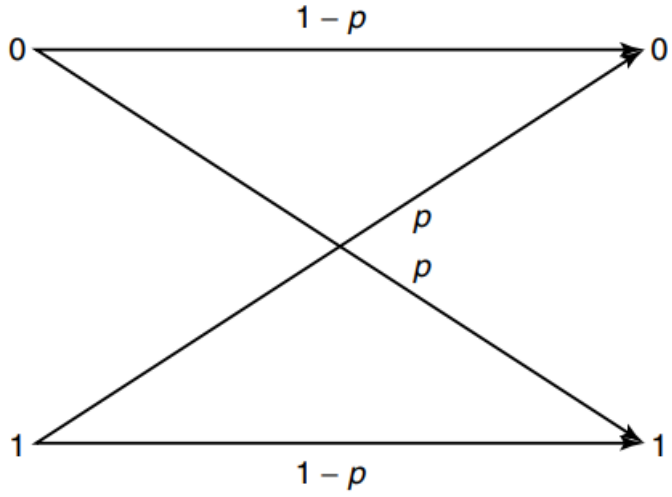


Fonte: Elaborado pelo autor

Nesta configuração não há erros: P(Y = 1|X = 1) = P(Y = 0|X = 0) = 1 e P(Y = 0|X = 1) = P(Y = 1|X = 0) = 0, para casos onde P(X = 1) = P(X = 0) = 0, 5, com isso é possível obter 1 bit de informação enviado, sendo essa a capacidade do canal.

Exemplo 4: capacidade de canal de um canal binário simétrico (BSC).

Figura 4 – Canal Simétrico Binário

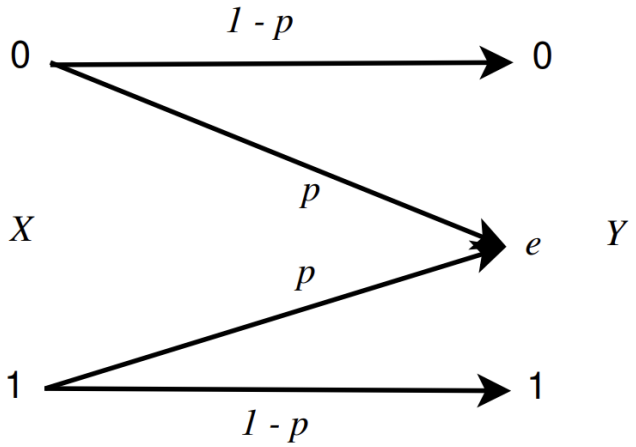


Fonte: Elaborado pelo autor

Nesta configuração, quando há erros, um 0 é recebido como 1, bem como 1 pode ser recebido como 0. Para esse canal P(Y = 1|X = 1) = P(Y = 0|X = 0)= 1-p e P(Y = 0|X = 1) = P(Y = 1/X = 0) = p. A capacidade desse canal é C = 1-H(p), sendo H(p) a entropia de uma fonte binaria. Sendo assim caso H(p) = 0, se torna um canal ideal, mas caso H(p) = 1, a capacidade de canal se torna 0 e a informação é completamente perdida.

Exemplo 5: Fig. 5 ilustra o Canal Binário com Apagamento (Binary Erasure Channel - BEC). Nesse tipo de canal, existem duas possíveis entradas, mas a saída apresenta uma terceira possibilidade, denominada apagamento, que indica um bit corrompido durante a transmissão pelo canal. Essa terceira saída é representada por e, e a probabilidade de sua ocorrência é expressa como $P[Y = e|X = 1] = P[Y = e|X = 0] = p$.

Figura 5 – Capacidade de Canal do BEC



Fonte: Elaborada pelo autor

A capacidade desse canal é dada por $C = 1 - p$. Nesse tipo de canal, quando um bit 0 ou 1 é recebido, há a certeza de que foi corretamente transmitido. Caso ocorra um apagamento, ele pode ser identificado e o bit correspondente pode ser retransmitido.

2.2 Código de Blocos Lineares

Uma informação a ser transmitida ou armazenada digitalmente, por razões práticas, é codificada em dígitos binários, 0 e 1, portanto, nesse tópico serão discutidos os códigos de bloco com símbolos do corpo binário GF(2).

Em um codificador de bloco, a sequência de informação binária é segmentada em blocos de mensagens com k bits, denotados por u = (u_0, u_1, . . . , u_{k-1}), onde $ui \in GF(2)$, i = 0, 1, . . . , k-1, assim teremos 2k possíveis mensagens. Cada mensagem u é transformada em uma palavra-código v com n bits. Os n-k bits introduzidos na mensagem u são chamados bits de verificação de paridade que são a redundância utilizada para o decodificador identificar se houve erros durante a transmissão e, se possível, corrigi-los. Seja K um corpo finto com q elementos. Temos, portanto, para cada número natural n, um K-espaço vetorial K^n de dimensão n.

2.2.1 Definição:

Um código de bloco de comprimento n e 2^k palavras código é dito código linear $C \subset k^n$, denotado por (n, k), quando C for subespaço de dimensão k de k^n. Seja (n, k) um código linear e seja ($g_0, g_1, ..., g_{k-1}$) uma de suas bases, portanto, todo elemento do código se escreve de modo único na forma:

$$v = u_0 g_0 + u_1 g_1 + ... + u_{k-1} g_{k-1} \tag{2.13}$$

Onde u_i, i = 0, 1,..., k-1 são coordenadas da mensagem $u = (u_0, u_1, ..., u_{k-1})$ de comprimento k.

Os vetores da base $(g_0, g_1, ..., g_{k-1})$, formam a matriz geradora do código G de ordem k x n.

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix} \tag{2.14}$$

Onde $g_i = (g_{i0}, g_{i1}, ..., g_{i,n-1})$ para $0 \leq i < k$. Seja K um corpo finito. Considere a transformação linear definida por:

$$T : K^k \to K^n \tag{2.15}$$

$$u \to u * G \tag{2.16}$$

Se u = $(u_0, u_1, ..., u_{k-1})$ é a mensagem original, então cada palavra código será dada por:

$$v = u * G \tag{2.17}$$

$$v = (u_0, u_1, \ldots, u_{k-1}) \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix} \tag{2.18}$$

$$v = u_0 * g_0 + u_1 g_1 + ... + u_{k-1} * g_{k-1} \tag{2.19}$$

Exemplo 5: Considere o corpo finito K = GF(2). O código de bloco linear $C \subset GF(2)^7$ de dimensão 4, denotado por (7, 4), sendo chamado de código de Hamming, cuja matriz geradora é dada por:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1101000 \\ 0110100 \\ 1110010 \\ 1010001 \end{bmatrix}$$

Seja u = (1101) a mensagem de entrada no codificador, então a palavra-código correspondente será:

$$v = u * G$$

$$v = (1101) * \begin{bmatrix} 1101000 \\ 0110100 \\ 1110010 \\ 1010001 \end{bmatrix}$$

$$v = (0001101)$$

Os códigos de Hamming formam uma classe de códigos lineares da forma $(2^m - 1, 2^m - 1 - m)$, para m > 2, que foram desenvolvidos em 1950 por Richard W. Hamming. Tais códigos são bastante eficientes na correção de erros simples, ou seja, corrigem apenas um único erro.

Os códigos de bloco lineares cujos k bits da mensagem original permanecem inalterados na palavra-código são chamados de códigos sistemáticos. É possível observar um código de bloco sistemático no Exemplo 5.

$$v = (1101) \rightarrow v = (000\mathbf{1101})$$

Um código de bloco linear sistemático (n,k) é completamente determinado pela matriz geradora G de ordem k x n da forma padrão:

$$G = \begin{bmatrix} p_{00} & p_{01} & \dots & p_{0,n-k-1} & |1 & 0 & \dots & 0 \\ p_{10} & p_{11} & \dots & p_{1,n-k-1} & |1 & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & |\dots & \dots & \dots & \dots \\ p_{k-1,0} & p_{k-1,1} & \dots & p_{k-1,n-k-1} & |1 & 0 & \dots & 0 \end{bmatrix} \tag{2.20}$$

Onde $p_{ij} \in GF(2)$, $\forall 0 \leq i < k$ e $\forall 0 \leq j < n - k$. Assim,

$$G = [P|I_k]$$

Onde I_k é a matriz identidade k x k e $P = (p_{ij})_{k\ x\ n-k}$, $p_{ij} \in GF(2)$. A partir de G, com algumas manipulações algébricas, é possível uma matriz $H = [I_{n-k}P^T]$, chamada de matriz de verificação de paridade, chamada simplesmente de matriz de paridade.

A matriz de paridade H é utilizada no processo de detecção e correção de erros, pois dada uma n-upla v é possível afirmar que v é uma palavra código de um código (n, k) gerado pela matriz $G = [P|I_K]$ se, e somente se, $v.H^T = 0$. Portanto, a partir da matriz H podemos identificar quando uma mensagem v pertence ou não ao código.

Considere um código linear (n,k) com matriz geradora G e matriz de paridade H. Seja $v = v_0, v_1, ..., v_{n-1})$ uma palavra código transmitida por um canal ruidoso e $r = (r_0, r_1, ..., r_{n-1})$ um vetor recebido no decodificador. Devido aos ruídos do canal de transmissão o vetor recebido r pode ser diferente de v. Essa diferença é o padrão de erro dado por:

$$e = r + v = (e_0, e_1, .., e_{n-1}) \tag{2.21}$$

Note que $e_i = 1$ implica que houve erro na i-ésima coordenada e, caso contrário, teremos $e_i = 0$, porque são consideradas as operações no corpo de Galois GF(2). Além disso, podemos reescrever $r = v + e$. No processo de decodificação, ao receber uma mensagem r o decodificador precisa primeiramente determinar se existem ou não erros de transmissão, para isto, o decodificador calcula a síndrome de r:

$$S = r * H^T = (s_0, s_1, .., s_{n-k-1}) \tag{2.22}$$

Se r é uma palavra código, então $S = r * H^T = 0$. Nesse caso não existem erros, ou seja, e = 0. No entanto, a recíproca não é verdadeira. É possível haver erros que não possam ser detectados, isto é, s = 0 mas r não é a palavra código transmitida. Isto ocorre quando ocorre um padrão de erro não-nulo e pertence ao código. Nesse caso r = v + e, é a soma de duas palavras código e temos $S = r * H^T = 0$. Portanto, estamos diante de um erro de decodificação.

Podemos afirmar que $s \neq 0 \leftrightarrow r$ contém algum erro. Além disso, um fato importante que podemos observar é que a síndrome depende apenas do padrão de erro, e não da palavra-código transmitida, pois $s * H^T = (v + e) * H^T = v * H^T + e * H^T$, mas como v é palavra-código então $v * H^T = 0$. Logo, $s = e * H^T$.

Assim, é observado que é possível detectar erros introduzidos no canal de transmissão. Contudo, nem todos os padrões de erro podem ser corretamente decodificados.

Exemplo 6 Considere o código de Hamming (7, 4) dado pela Tabela 1 cuja matriz geradora é dada por:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{bmatrix} = \begin{bmatrix} 1101000 \\ 0110100 \\ 1110010 \\ 1010001 \end{bmatrix}$$

Tabela 1 – Código de Hamming (7,4)

Mensagens	**Palavras Códigos**
0 0 0 0	0 0 0 0 0 0 0
1 0 0 0	1 1 0 1 0 0 0
0 1 0 0	0 1 1 0 1 0 0
1 1 0 0	1 0 1 1 1 0 0
0 0 1 0	1 1 1 0 0 1 0
1 0 1 0	0 0 1 1 0 1 0
0 1 1 0	1 0 0 0 1 1 0
1 1 1 0	0 1 0 1 1 1 0
0 0 0 1	1 0 1 0 0 0 1
1 0 0 1	0 1 1 1 0 0 1
0 1 0 1	1 1 0 0 1 0 1
1 1 0 1	0 0 0 1 1 0 1
0 0 1 1	0 1 0 0 0 1 1
1 0 1 1	1 0 0 1 0 1 1
0 1 1 1	0 0 1 0 1 1 1
1 1 1 1	1 1 1 1 1 1 1

Se u = (1011) é uma mensagem a ser codificada, a palavra-código é dada por:

$$v = u * G$$

$$v = (1101) * \begin{bmatrix} 110|1000 \\ 011|0100 \\ 111|0010 \\ 101|0001 \end{bmatrix}$$

$$v = (0001101)$$

Note que a matriz G está na forma sistemática:

$$G = [P|I_4]$$

Logo, a matriz de paridade pode ser facilmente encontrada, pois sabemos que é da forma:

$$H = [I_3|P^T]$$

$$H = \begin{bmatrix} 100|1011 \\ 010|1110 \\ 001|0111 \end{bmatrix}$$

Seja v = (1001011) a palavra-código transmitida e r = (1001001) o vetor recebido. Para determinar se houve erro na transmissão, o decodificador calcula a síndrome de r, dada por:

$$S = r * H^T$$

$$s = (1001001) * \begin{bmatrix} 100 \\ 010 \\ 001 \\ 110 \\ 011 \\ 111 \\ 101 \end{bmatrix} = \Big(111\Big)$$

Como $s \neq 0$, então r não pertence ao código de bloco linear (7,4) dado pela Tabela 1. Para encontrar o padrão de erro basta usar o fato que $s = e * H^T$ e é mostrado que:

$$(111) = (e_0e_1e_2e_3e_4e_5e_6) * \begin{bmatrix} 100 \\ 010 \\ 001 \\ 110 \\ 011 \\ 111 \\ 101 \end{bmatrix} = \Big(e_0 + e_3 + e_5 + e_6 \quad e_1 + e_3 + e_4 + e_5 \quad e_2 + e_4 + e_5 + e_6\Big)$$

O padrão de erro com menor número de coordenadas diferente de zero, ou seja, que tem o menor número de erros e que satisfaz a equação acima é o vetor e = (0000010). Considerando um canal binário simétrico (BSC), no qual a probabilidade de transmitir um dígito errado é igual para os dígitos 1 e 0, então o vetor e = (0000010) é o vetor erro mais provável.

Logo, a palavra-código decodificada será:

$$v' = r + e$$

$$v' = (1001001) + (0000010)$$

$$v' = 1001011$$

2.3 Polarização de Canal

A polarização de canal é de extrema importância na separação entre os canais bons e os canais ruins de transmissão, sendo os canais bons utilizados pelos bits de informação e os canais ruins, onde não há pouca ou nenhuma informação mútua, sendo constituídos pelos chamados bits congelados. Para a esquematização da técnica de polarização de canal, Arikan propôs o uso de vários B-DMC(*Binary Discrete Memoryless Channel*) conectados entre si, se valendo da recursividade.

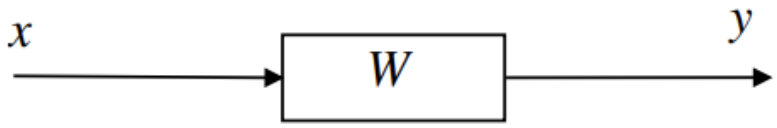
A capacidade dos canais BEC em conectados podem ser calculados de maneira recursiva pelas seguintes fórmulas:

$$I(W_N^{(2i-1)}) = I(W_{N/2}^{(i)})^2 \tag{2.23}$$

$$I(W_N^{(2i)}) = 2 * I(W_{N/2}^{(i)}) - I(W_{N/2}^{(i)})^2 \tag{2.24}$$

Como exemplo de um B-DMC, é mostrado um BEC(*Binary Eraser Channel*) na Fig. 6.

Figura 6 – BEC com N = 1

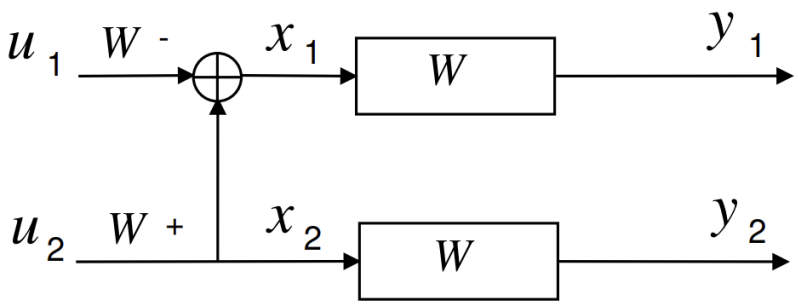


Fonte:(TERÇAS., 2021)

Nesse caso inicial, com $N = 1$ e considerando $I(W_1^{(1)})$ = 1-p, é calculado que: I(W) = 1-p = 0.5

Para $N = 2$, na Fig. 7, e considerando com $I(W_1^{(1)})$ = 1-p, temos que:

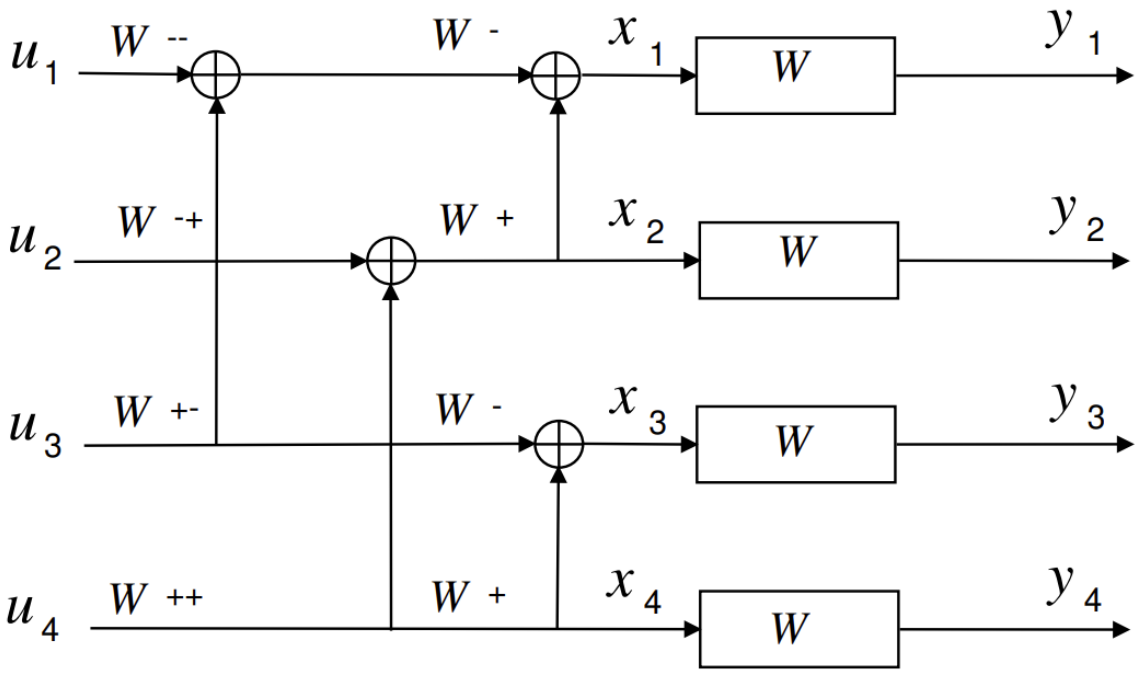
Figura 7 – BEC com N = 2



Fonte:(TERÇAS., 2021)

$$I(W^-) = I(W)^2 = (0.5)^2 = 0.25,$$
$$I(W^+) = 2 \cdot I(W) - I(W)^2 = 2 \cdot (0.5) - (0.5)^2 = 0.75.$$

Figura 8 – BEC com N = 4



Fonte:(TERÇAS., 2021)

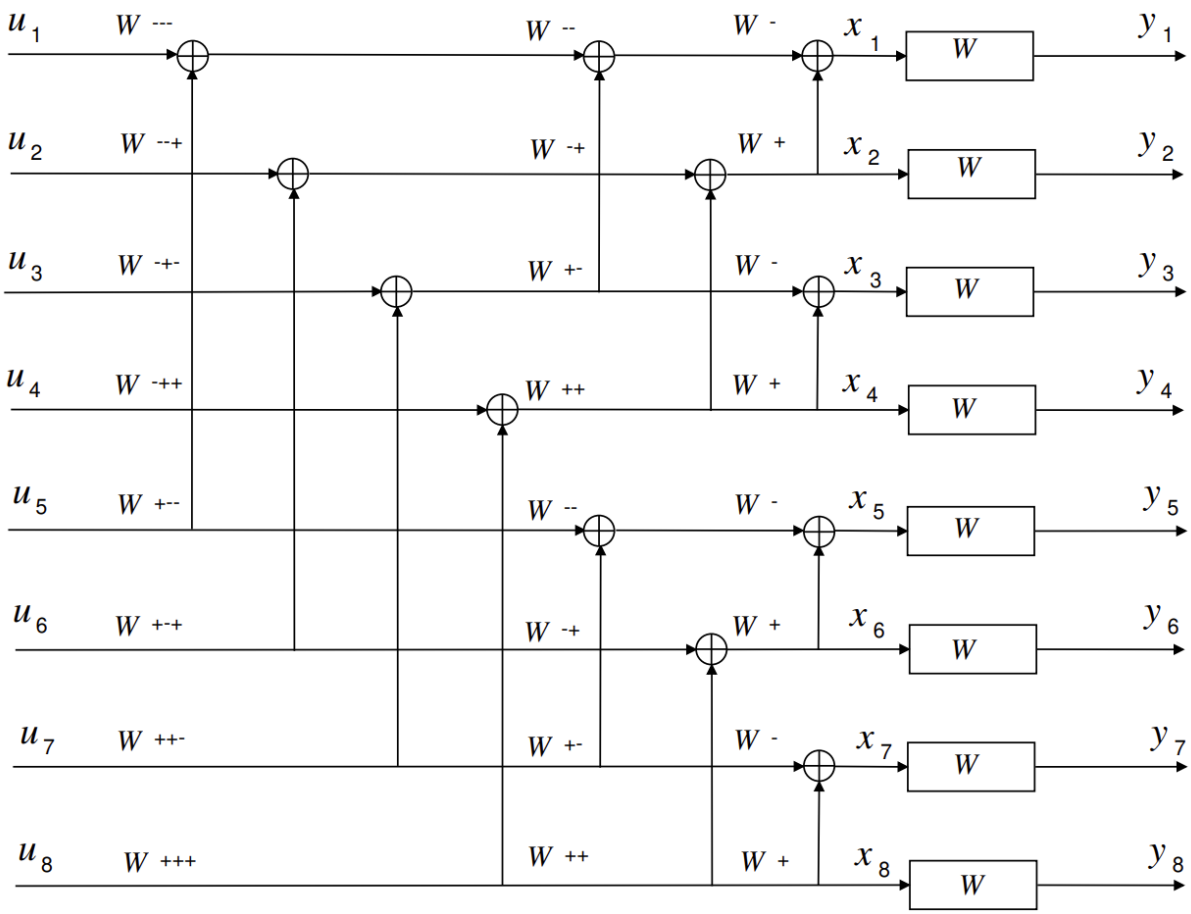
Para a Fig. 8 com $N = 4$, é calculado que:

$$I(W^{--}) = I(W^{-})^2 = (0.25)^2 = 0.0625,$$
$$I(W^{-+}) = 2 \cdot I(W^{-}) - I(W^{-})^2 = 2 \cdot (0.25) - (0.25)^2 = 0.4375,$$
$$I(W^{+-}) = I(W^{+})^2 = (0.75)^2 = 0.5625,$$
$$I(W^{++}) = 2 \cdot I(W^{+}) - I(W^{+})^2 = 2 \cdot (0.75) - (0.75)^2 = 0.9375.$$

É notório ver que a $u1$ tende ao extremo 0, se tornando um canal ruim para a transmissão, enquanto o $u4$ tende ao extremo 1, se tornando o melhor canal para a transmissão de um bit de informação.

$N = 8$ Neste caso a capacidade será novamente dividida entre 8 canais, como na Fig. 9

Figura 9 – BEC com N = 8



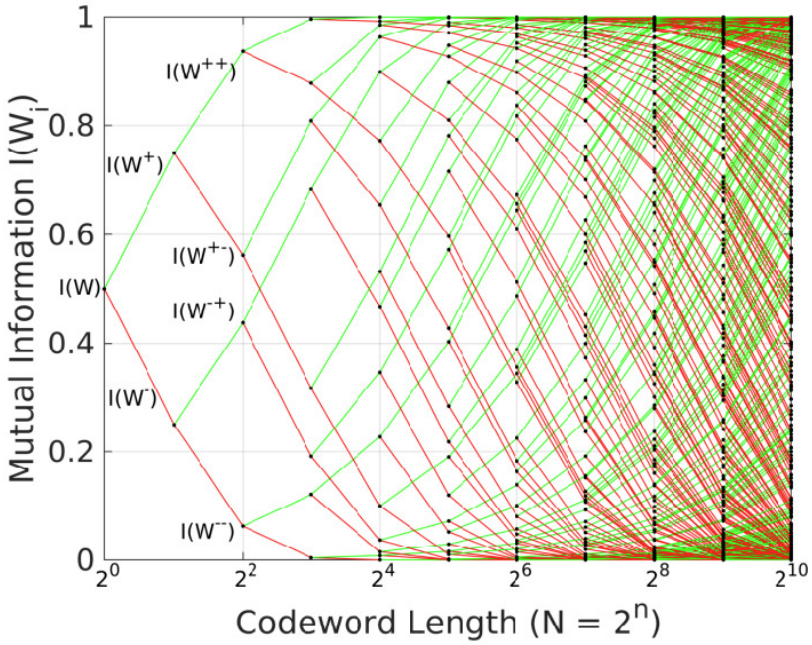
Fonte: (TERÇAS., 2021)

$$I(W^{---}) = I(W^{-})^2 = (0.0625)^2 = 0.0039$$
$$I(W^{--+}) = 2I(W^{-}) - I(W^{-})^2 = 2(0.0625) - (0.0625)^2 = 0.1210$$
$$I(W^{-+-}) = I(W^{+})^2 = (0.4375)^2 = 0.1914$$
$$I(W^{-++}) = 2I(W^{+}) - I(W^{+})^2 = 2(0.4375) - (0.4375)^2 = 0.6835$$
$$I(W^{+--}) = I(W^{+})^2 = (0.5625)^2 = 0.3164$$
$$I(W^{+-+}) = 2I(W^{+}) - I(W^{+})^2 = 2(0.5625) - (0.5625)^2 = 0.8085$$
$$I(W^{++-}) = I(W^{+})^2 = (0.9375)^2 = 0.8789$$
$$I(W^{+++}) = 2I(W^{+}) - I(W^{+})^2 = 2(0.9375) - (0.9375)^2 = 0.9960$$

Neste caso, para uma transmissão de 4 bits de informação poderíamos congelar as entradas u_1 , u_2 ,u_3 e u_5 e transmitir as informações pelas entradas u_4 , u_6 , u_7 e u_8.

Na Figura 10 é demonstrado que os canais polarizados tendem aos extremos, 0 ou 1, conforme o tamanho N do canal tende ao infinito.

Figura 10 – Gráfico da polarização de canais



Fonte: (AL., 2019)

2.4 Códigos Polares

Os códigos polares foram propostos e desenvolvidos por Arikan (ARIKAN, 2009), os mesmos são um tipo de código de blocos lineares que utilizam o princípio da polarização de canal para que os bits de informação sejam enviados através dos canais confiáveis (bons)

e os bits congelados sejam enviados para os canais mais ruidosos (ruins e não confiáveis). Nesse tópico será demonstrado como funciona a codificação e a polarização por meio de códigos polares.

2.4.1 Codificação de Códigos Polares

Como visto anteriormente os códigos polares induzem um conjunto de canais bons e ruins, para que os bits de informação possam ser transmitidos através dos bons canais induzidos, enquanto a entrada para os canais ruins induzidos são congeladas. Quando a palavra-código de comprimento N é infinitamente longa, os canais são polarizados em k canais bons (ou os chamados canais perfeitos) e $(N - k)$ canais ruins (ou os chamados canais inúteis). Assim um código polar codifica k bits de informação em N bits codificados usando $(N - k)$ bits redundantes, que são chamados de "bits congelados". É caracterizada pelos parâmetros (N, k, F, u_f) , onde $F \subset 1, 2, ..., N$ especifica a localização dos bits congelados, enquanto u_f é um vetor de bits $(N - k)$ de bits congelados, que são conhecido pelo decodificador. O desempenho dos códigos polares depende nos parâmetros N, k, bem como F. Em particular, F é específico do canal e deve ser otimizado para o canal sob consideração. No entanto, o desempenho dos códigos polares não é afetado pelo valor de bits congelados, mais precisamente o vetor u_f , se o canal for simétrico. Geralmente, u_f é assumido como um vetor totalmente zero. É importante notar aqui que os códigos polares são intrinsecamente compatíveis com a taxa, uma vez que a taxa de codificação pode ser variada simplesmente alterando o número de bits congelados, usando o mesmo codificador G_N. As localizações F dos bits congelados podem ser selecionadas usando uma sequência que lê os locais. As melhores localizações para uma taxa de codificação são tipicamente um subconjunto das melhores localizações para qualquer taxa de codificação mais baixa.

Um codificador arbitrário G_N pode ser representado como:

$$x_1^N = u_1^N G_N \tag{2.25}$$

Para o código polar de Arikan, G_N é o enésimo produto de Kronecker de a matriz kernel (2×2). Mais especificamente, o Arikan kernel G_N pode ser representado em forma de matriz da seguinte forma:

$$G = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} \tag{2.26}$$

enquanto o codificador de N bits G_N é definido recursivamente como:

$$G = G_2^{\otimes n} \begin{pmatrix} G_{N/2} & 0 \\ G_{N/2} & G_{N/2} \end{pmatrix} \tag{2.27}$$

Assim, o código polar de Arikan tem uma estrutura recursiva que invoca $n = log_2 N$ camadas de polarização e cada camada de polarização usa portas $N/2$ XOR. Assim, a operação de codificação da equação 2.27 impõe uma complexidade de $O(Nlog_2 N)$. Também podemos notar de Eq. 2.27 que os códigos polares assumem uma estrutura não sistemática. Mais tarde, códigos polares sistemáticos foram derivados em (ARIKAN., 2011) (SARKIS; GIARD., 2016) (VANGALA; VITERBO., 2015) (CHEN; ZHANG., 2016), que superou os códigos polares não sistemáticos clássicos em termos do BER, mantendo o mesmo *Block Error Ratio (BLER)* e codificação, e complexidade de decodificação. Restringimos as discussões para os códigos polares não sistemáticos clássicos neste relatório.

O codificador polar de 8 bits pode ser formulado da seguinte forma:

$$G_8 = G_2^{\otimes 3} \begin{pmatrix} G_2 & 0 & 0 & 0 \\ G_2 & G_2 & 0 & 0 \\ G_2 & 0 & G_2 & 0 \\ G_2 & G_2 & G_2 & G_2 \end{pmatrix}$$

$$G_8 = G_2^{\otimes 3} \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{pmatrix} \tag{2.28}$$

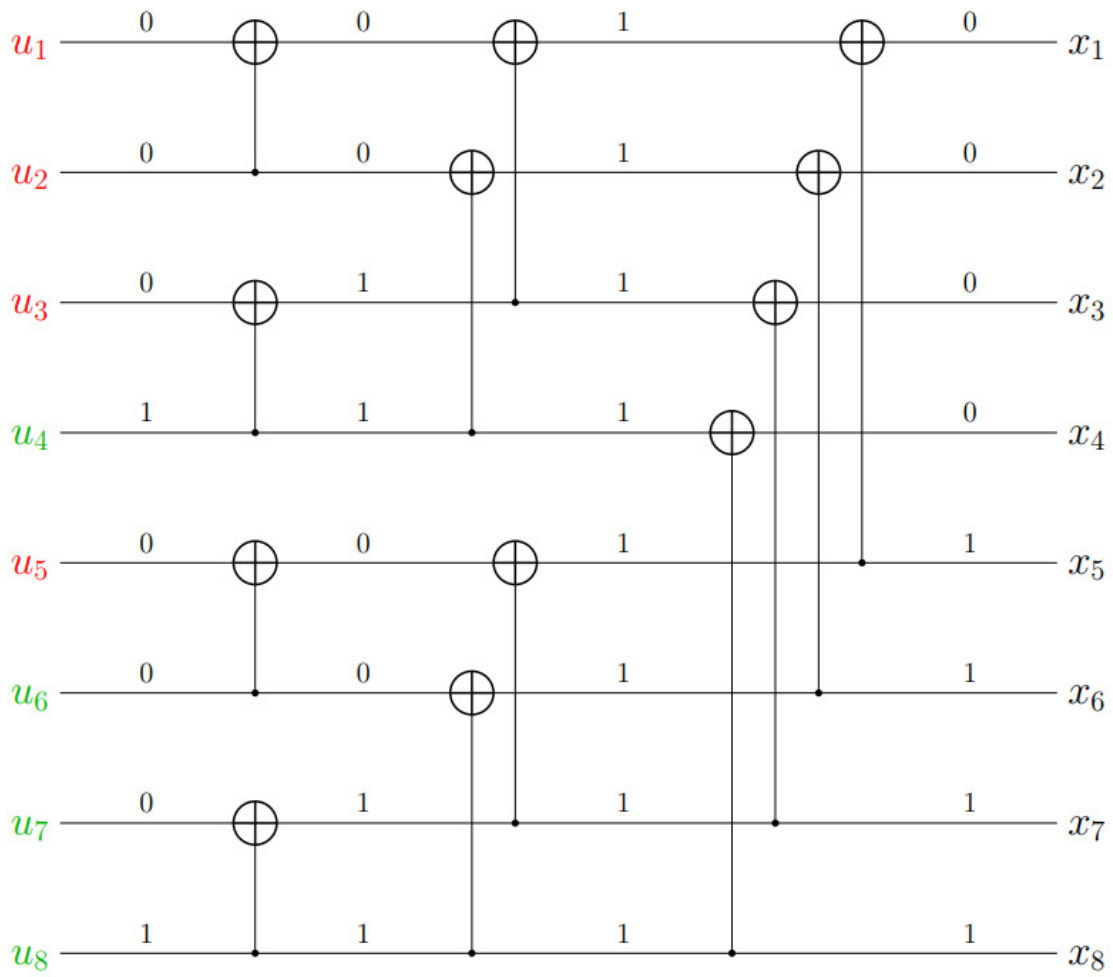
Consideremos um código polar de 8 bits tendo $k = 4, f = 1, 2, 3, 5, u_f = (0000)$ e uma sequência de bits de informação $uf_c = (1001)$. Então a saída codificada pode ser calculada do seguinte modo:

$$\begin{aligned} x_1^8 &= (0 \quad 0 \quad 0 \quad u_4 \quad 0 \quad u_6 \quad u_7 \quad u_8) \times G_8 \\ &= (0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 1) \times G_8 \\ &= (0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1) \end{aligned} \tag{2.29}$$

Observe que f_c denota o conjunto complementar de f, que especifica a localização dos bits de informação. A palavra-código da Eq.2.29 também pode ser elaborado diretamente a partir da codificação circuito, como exemplificado na Figura 11. O processo de codificação da Eq. 2.25 pode ser reformulado como:

O processo de codificação da Eq. 2.25 pode ser reformulado como:

Figura 11 – Exemplo do processo de codificação polar



Fonte: (AL., 2019)

$$x_1^N = u_{fc} G_N(f_c) + u_f G_N(f) \tag{2.30}$$

onde $G_N(f)$ é uma submatriz de G_N contendo apenas as linhas com índices em f. Quando u_f é definido para uma sequência de bits totalmente zero, Eq. 2.30 reduz-se a:

$$x_1^N = u_{fc} G_N(f_c) \tag{2.31}$$

Onde $G_N(f_c)$. é uma matriz geradora (kN). Assim, polares códigos são equivalentes a códigos de bloco linear com uma matriz geradora $G_N(f_c)$. O codificador de códigos polares traz consigo a benefício adicional de escalabilidade, tanto em termos de comprimento de palavra código, bem como a taxa de codificação. Mais especificamente, o recurso de escalabilidade de comprimento vem da natureza recursiva de G_N, enquanto a taxa pode ser modificada apenas alterando o número de pedaços congelados. Também pode ser notado na Eq. 2.30 que quando os bits congelados não são definidos para uma sequência de zeros,

então o código resultante é um coset do código de bloco linear com a matriz geradora $G_N(f_c)$ e o coset é determinado pelo vetor $u_f G_N(f)$.

Os códigos polares estão intimamente relacionados com a família de RM códigos (ARIKAN., 2008) , uma vez que ambos contam com o codificador G_N de Eq. 2.27. Explicitamente, dado um par de inteiros $0 \leq r \leq n$, existe um código RM de comprimento de palavra de código $N = 2^n$ e comprimento da palavra de informação $k = \sum_{i=0}^{r} \binom{n}{i}$, cujo gerador a matriz G_{RM} é uma submatriz de G_N. Esta propriedade é análoga ao dos códigos polares. No entanto, enquanto a matriz geradora $G_N(f_c)$ de um código polar corresponde às linhas mais confiáveis de G_N , a matriz geradora G_{RM} de um código RM consiste em linhas de G_N com pesos de Hamming $\geq 2^{m-r}$.

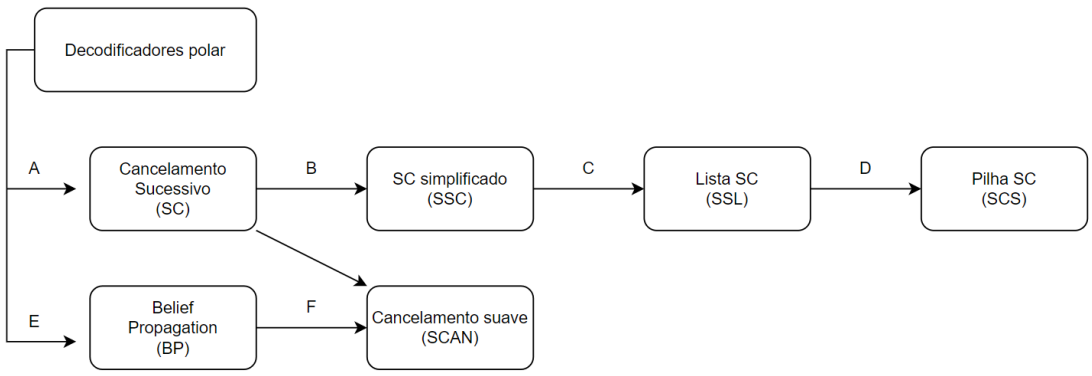
Equivalentemente, podemos dizer que congelamos os canais menos confiáveis dos códigos polares, enquanto congelamos o menor peso de Hamming canais em códigosRM. Consequentemente, os códigos polares aproximam-se de capacidade de shannon, enquanto os códigos RM exibem uma distância mínima alta. É interessante ressaltar que a seleção baseada em confiabilidade de canais de bits congelados em um BEC coincide com o menor Canais de peso de Hamming para $n = 3$ e $n = 4$. Assim, o código polar da Eq. 2.29 é equivalente ao código $RM(8, 4, 4)$.

No entanto, os benefícios dos códigos polares começam a surgir à medida que aumentamos N (ARIKAN., 2008).

2.4.2 Decodificação de Códigos Polares

Desde o início dos códigos polares, intensos esforços de pesquisa têm investido na melhoria dos algoritmos de decodificação polar tanto do ponto de vista algorítmico como do ponto de vista da perspectiva de implementações de hardware. Nesta projeto foi estudado o algoritmo de cancelamento sucessivo(SC) identificado na Fig. 12.

Figura 12 – Algoritmos de decodificação polar.

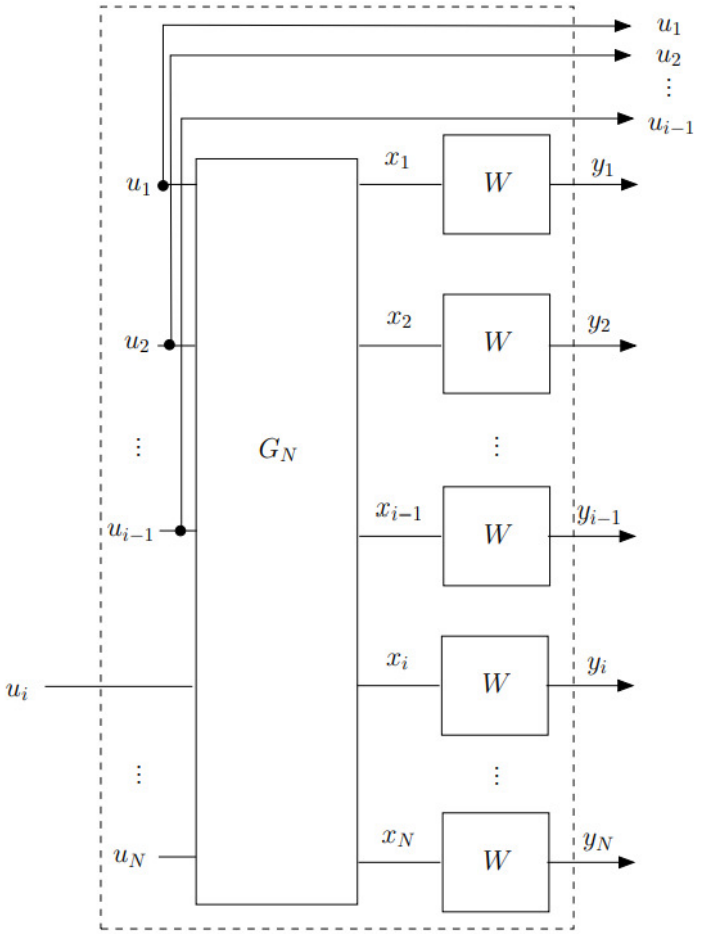


Fonte: (AL., 2019)

2.4.2.1 Decodificador de cancelamento sucessivo

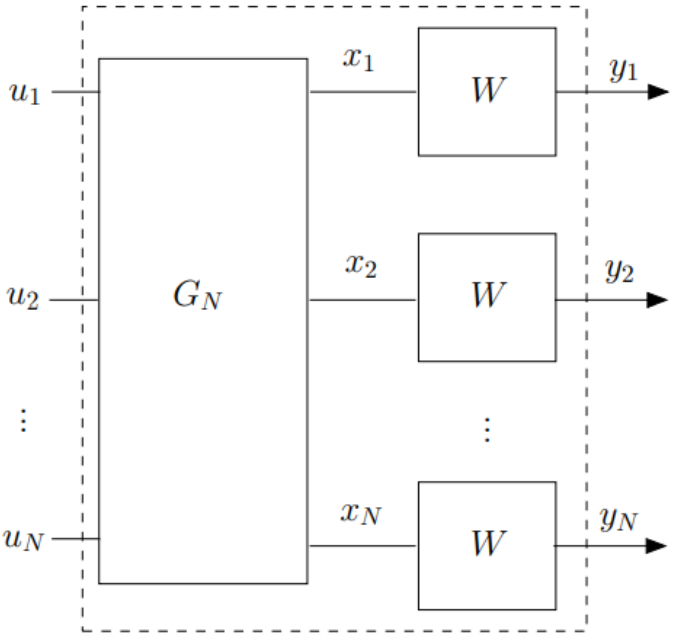
O artigo inicial de Arikan (ARIKAN, 2009) propôs o decodificador polar de Cancelamento Sucessivo (SC) baseado em *Likelihood Ratio(LR)*. Pela Fig. 13 que o canal composto W^N pode ser dividido em N canais polarizados de bits tais que o i-ésimo canal de bits W_i pega a entrada u_i e produz a saída $y_1^N u_i^{i-1}$. O canal associado com probabilidades de transição são denotadas por $P_i(y_1^N, u_1^{(i-1)}|u_i)$, que pode ser estimado usando um decodificador SC. Para elaborar, o processo de combinação de canais da Fig. 14 acopla o bits de entrada u_i^N. Assim, um decodificador SC reverte esse processo no receptor removendo a contribuição, ou mais precisamente, interferência dos bits u_1^{i-1} dos bits codificados recebidos y_1^N, portanto, revelando o valor do i-ésimo bit u_i.

Figura 13 – Divisão de canais: o canal de bits induzido W_i



Fonte: (AL., 2019)

Figura 14 – Combinação de canais: o codificador G_N combina o N usos de um B-DMC para construir um canal composto W_N



Fonte: (AL., 2019)

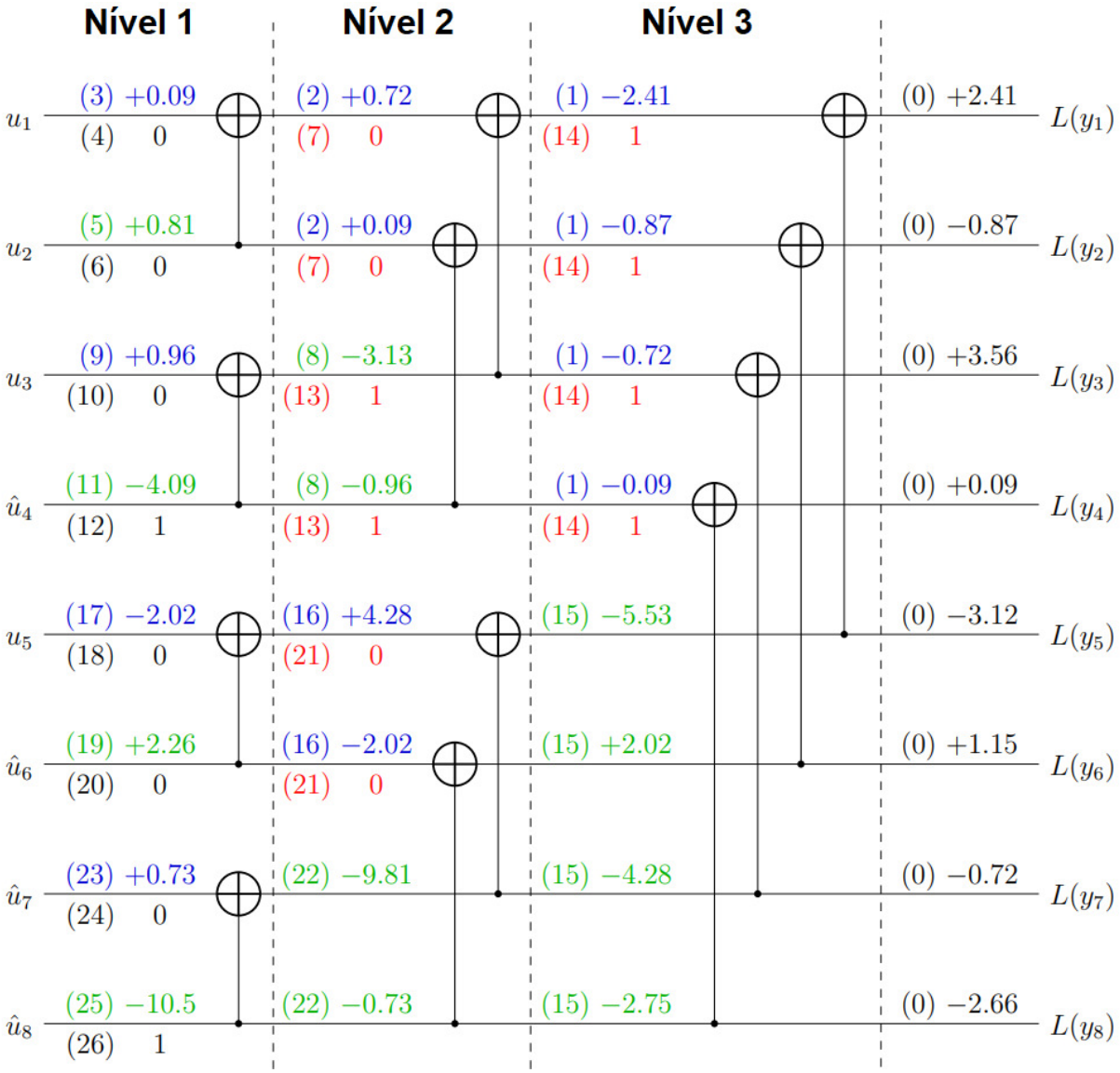
Um decodificador SC opera no mesmo circuito que o do codificador, como exemplificado na Fig. 15, para o codificador polar de Fig. 11. No entanto, enquanto um codificador sempre processa os bits da esquerda para a direita, um decodificador SC opera da direita para a esquerda assim como da esquerda para a direita. Para elaborar, um decodificador SC executa cálculos relativos aos XORs no circuito de acordo com uma sequência que é ditada pela disponibilidade de dados no lado esquerdo e direito do XOR, que introduz dependências de dados no processo de decodificação. Por isso, a funcionalidade de cada XOR no circuito de decodificação varia, ao realizar operações em LLRs em diferentes etapas do Processo de decodificação SC.

Na Fig. 15: Exemplo do processo de decodificação SC: O código polar $N = 8$ tendo $k = 4, f = 1, 2, 3, 5$ e $u_f = (0000)$ é usado para decodificar os LLRs codificados recebidos $L(y_i)$ no $k = 4bits$ de informação recuperados $\hat{u}_{f_c} = (1001)$. Os LLRs obtido usando as funções f e g da Eq. 2.32 e Eq. 2.35 são mostrados acima de cada conexão em azul e verde, respectivamente. Os bits obtidos usando os cálculos de soma parcial de Eq. 2.36 e Eq. 2.37 são mostrados abaixo de cada conexão em vermelho. Os números que acompanham entre parênteses identificam a etapa de o processo de decodificação SC onde o LLR ou bit correspondente torna-se disponível.

O LLR L(b) referente ao bit b é definido como:

$$L(b) = log\left(\frac{P(b=0)}{P(b=1)}\right)$$

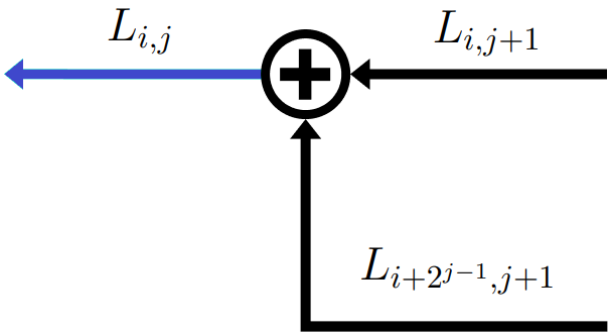
Figura 15 – Decodificador de cancelamento sucessivo.



Fonte: (AL., 2019)

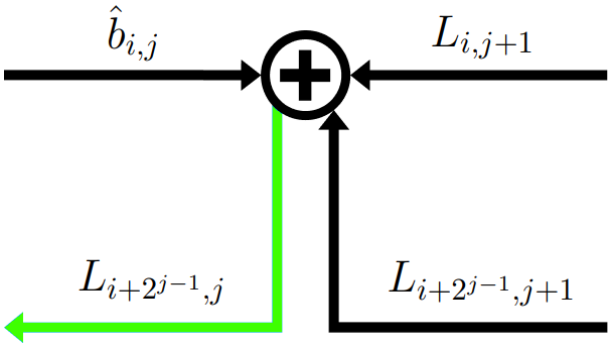
Existem três tipos de cálculos que pode ser realizados por um determinado XOR na decodificação, dependendo da disponibilidade de LLRs fornecidos nas conexões em seu lado direito, bem como na disponibilidade de bits fornecidos nas conexões em seu lado esquerdo. Vamos exemplificar isso considerando o elemento elementar de 2 bits do kernel da Fig. 15, que opera no i-ésimo e $(i + 2^{j-1})$ bits, onde $j \in [1, n]$ denota o índice de nível. A primeira ocasião em que um XOR pode contribuir para o processo de decodificação SC é quando um LLR foi fornecido por cada uma das conexões em seu lado direito, conforme mostrado na Fig. 16.

Figura 16 – Função $f(L_{i,j+1}, L_{i+2^{j-1},j+1})$: LLRs se propagam da direita para a esquerda



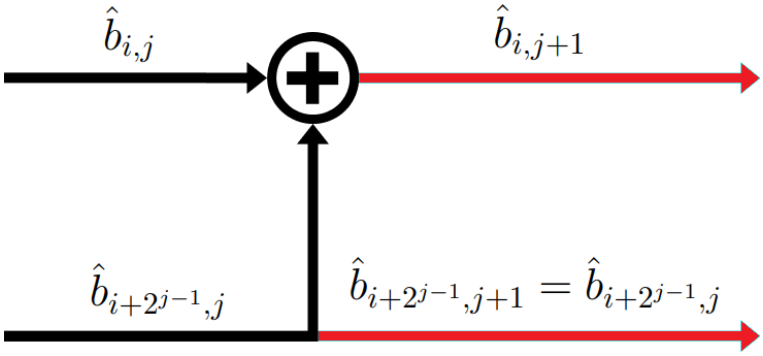
Fonte: Elaborado pelo autor

Figura 17 – Função $g(L_{i,j+1}, L_{i+2^{j-1},j+1}, \hat{b}_{i,j})$: mudar de propagação bits para propagar LLRs.



Fonte: Elaborado pelo autor

Figura 18 – Cálculo de soma parcial XOR $\hat{u}_1\hat{u}_2$: bits se propagam da esquerda para a direita



Fonte: Elaborado pelo autor

Como o XOR conecta o i-ésimo e $(i + 2^{j-1})$ bits, nos referimos ao primeiro e ao segundo desses dois LLRs como $L_{i,j+1}$e $L_{1+2^{j-1},j+1}$ respectivamente. Mais especificamente $L_{i,j+1}$ e $L_{1+2^{j-1},j+1}$ fornecer informações relativas ao bits $b_{i,j+1}$ e $b_{i+2^{j-1},j+1}$ respectivamente. Esses LLRs podem ser geradas pelo demodulador suave (para $j = n$) ou por os outros XORs no circuito (para $j < n$). Com base na entrada $L_{i,j+1}$ e $L_{i+2^{j-1},j+1}$, o XOR da Fig. 16 calcula o LLR $L_{i,j}$ para a primeira das duas conexões à esquerda, como segue:

$$f(L_{i,j+1}, L_{i+2^{j-1},j+1}) = L_{i,j+1}L_{i+2^{j-1},j+1} \tag{2.32}$$

onde o operador box-plus é definido como (HAGENAUER; PAPKE., 1996):

$$\begin{aligned} &L(b_1)L(b_2) \\ &= L(b_1 \oplus b_2) \\ &= ln\frac{1 + e^{(Lb_1)}e^{(Lb_2)}}{e^{(Lb_1)} + e^{(Lb_2)}} \\ &= 2tanh^{-1}(tanh(L(b_1)/2)tanh(L(b_2)/2) \end{aligned} \tag{2.33}$$

$$\begin{aligned} &sign(L(b_1))sign(L(b_2))min(|L(b_1)|, |L(b_2)|) \\ &+log(1 + e^{-|L(b_1)+L(b_2)|}) - log(1 + e^{-|L(b_1)-L(b_2)|}) \\ &sign(L(b_1))sign(L(b_2))min(|L(b_1)|, |L(b_2)|). \end{aligned} \tag{2.34}$$

Aqui, $L(b_1)$ e $L(b_2)$ são os LLRs pertencentes aos bits b_1 e b_2, respectivamente. O sinal$(\cdot)$ da Eq. 2.34 retorna -1 se seu argumento for negativo e $+1$ se seu argumento for positivo. Aqui, Eq. 2.34 é referida como a aproximação da soma mínima. Mais tarde no processo de decodificação SC, o bit estimado $\hat{b}_{i,j}$ é fornecido na primeira das conexões do lado esquerdo lado do XOR, como mostrado na Fig. 17.

Junto com o LLRs $L_{i,j+1}$ e $L_{i+2^{j-1},j+1}$ que foram fornecidos anteriormente usando as conexões do lado direito, permite que o XOR para calcule o LLR $L_{i+2^{j-1},j}$ para a segunda conexão em seu lado esquerdo, de acordo com a função g do seguinte modo:

$$\begin{aligned} L_{i+2^{j-1},j} &= g(L_{i,j+1}, L_{i+2^{j-1},j+1}, \hat{b}_{i,j}) \\ &= (-1)^{\hat{b}_{i,j}}L_{i,j+1} + L_{i+2^{j-1},j+1.} \end{aligned} \tag{2.35}$$

Podemos observar na Eq. 2.35 que a função g é análoga a operação de decodificação de um nó de repetição, uma vez que os dois LLR os valores são somados. Isso porque as informações pertencente ao bit $b_{i+2^{j-1},j}$ está contido bem como em $L_{i+2^{j-1},j+1}$. Além disso, o sinal de $L_{i,j+1}$ é invertido quando $\hat{b}_{i,j} = 1$, pois temos $b_{i+2^{j-1},j} = b_{i,j+1} \oplus b_{i,j}$.

Mais tarde ainda, o bit $\hat{b}_{i+2^{j-1},j}$ será fornecido na segunda conexão do lado esquerdo do XOR, conforme mostrado na Fig. 18. Juntamente com o bit $\hat{b}_{i,j}$ que foi previamente fornecido usando a primeira das conexões do lado esquerdo, isso permite o cálculo da soma parcial dos bits $\hat{b}_{i+2^{j-1},j+1}$ para a primeira e segunda conexão do lado direito do XOR, onde:

$$\hat{b}_{i,j+1} = XOR(\hat{b}_{i,j}, \hat{b}_{i+2^{j-1},j}), \tag{2.36}$$

$$\hat{b}_{i+2^{j-1},j+1} = \hat{b}_{i+2^{j-1},j}. \tag{2.37}$$

Como pode ser visto a partir das discussões acima, o função f da Eq. 2.32 pode ser usado para propagar LLRs da direita para a esquerda dentro do decodificador SC, enquanto os cálculos de soma parcial da Eq. 2.36 e Eq. 2.37 pode ser usado para propagar bits da esquerda para a direita e a função g da Eq. 2.35 pode ser usado para mudar da propagação de bits da esquerda para a direita para a propagação de LLRs da direita para a esquerda. O processo de decodificação SC começa processando LLRs da direita para a esquerda. No entanto, para que os LLRs possam ser propagados da direita para a esquerda, é necessário fornecer LLRs nas conexões na extremidade direita do circuito, ou seja, conexões à direita no nível 3 da Fig. 15.

No exemplo da Fig. 15, isso é realizado em o início do processo de decodificação SC, fornecendo sucessivas LLRs de um demodulador suave em conexões sucessivas na extremidade direita do circuito. Também podemos chamá-los LLRs de canal, uma vez que fornecem informações às saídas do canal. O processo de decodificação SC começa então usando a função f da Eq. 2.32 para propagar LLRs do borda direita do circuito de decodificação para a conexão superior na borda esquerda, permitindo que o primeiro bit seja recuperado (passos (0) a (4) na Fig. 15. Explicitamente, se o primeiro bit for um bit de informação, então uma decisão difícil é tomada com base no LLR resultante $L_{1,1}$. Por outro lado, se o primeiro bit for um bit congelado, então ele é definido como equivalente ao bit congelado conhecido. Então função g da Eq. 2.35 é usado para calcular o LLR referente para o segundo bit, revelando assim o seu valor (passos (5) e (6) na Fig. 15.

Em seguida, cada bit sucessivo de cima para baixo é recuperado usando os cálculos de soma parcial da Eq. 2.36 e Eq. 2.37 para propagar bits da esquerda para a direita, então usando a função g da Eq. 2.35 para um determinado XOR para mudar de propagação de bits para propagação LLR, antes de usar a função f para propagar LLRs para a próxima conexão na borda esquerda do circuito, permitindo que o bit correspondente seja recuperado. É pertinente mencionar aqui que se o bit na borda esquerda é um bit congelado, então o LLR associado é ignorada e o valor do bit é definido para o valor conhecido do bit congelado.

O processo de decodificação SC da Fig. 15 também pode ser visualizado sobre uma árvore de decodificação, como mostrado na Fig. 19. A árvore de decodificação de A Fig. 19 consiste em n = 3 níveis e cada nível é composto por 2^{n-i} nós pais e 2^{n-i+1} nós filhos; daí resultando $2^3 = 8$ nós folha no nível 1, que correspondem aos bits u. Para elaborado, o número de nós filhos em cada nível corresponde ao número de canais polarizados distintos

criados naquele nível. Obtemos dois tipos de polarização canais $W-$ e $W+$ no nível 3, que são ainda mais polarizados em $W--$, $W-+$, $W+-$ e $W++$ no nível 2 e depois em oito tipos no nível 1. Isso resulta em 2^{n-i+1} tipos de canais polarizados em cada nível. Além disso, a árvore de decodificação no nível 3 começa com um nó pai de comprimento N, cujo comprimento reduz pela metade em cada nó filho, portanto, adotando uma abordagem recursiva de dividir e conquistar.

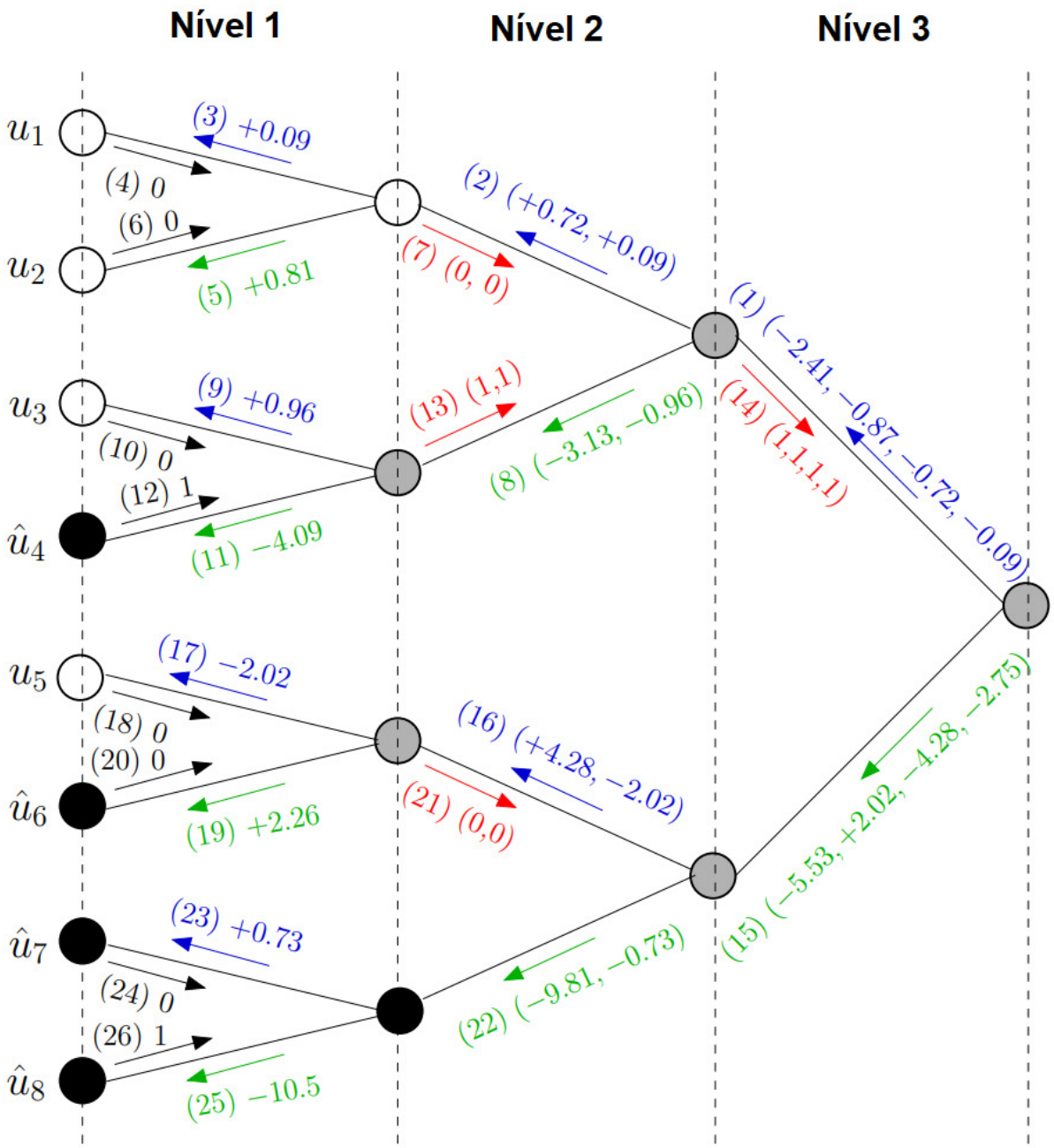
Vamos agora elaborar o fluxo de LLRs e bits através a árvore de decodificação da Fig. 19, onde cada nó atua como um decodificador local executando as operações XOR da Fig. 16,17,18. O SC processo de decodificação começa com o nó pai no nível 3, que tem o canal LLRs do demodulador suave. Este nó pai gera LLRs para o nó filho superior usando a função F da Fig. 16. Em seguida, ele espera até receber o código decodificado bits do nó filho superior e transfere o controle parental para o nó filho superior, que agora atua como o próximo nó pai. Este processo continua de forma recursiva até chegarmos a os nós folha no nível 1. Neste ponto, uma decisão difícil é tomada pertencente aos bits não codificados u e enviados para o nó pai em nível 1. Ao receber os bits decodificados, o nó pai no o nível 1 calcula os LLRs para o nó filho inferior usando o g função da Fig. 17 e aguarda os bits decodificados de o nó filho inferior. Ao receber os bits, o nó pai no nível 1 executa a operação XOR da Fig. 18 e envia os bits resultantes para seu nó pai no Nível 2. O processo continua recursivamente até que todos os bits não codificados u tenham sido recuperados.

A Fig. 19 mostra árvore de decodificação SC para o circuito de decodificação SC de Fig. 15. Os nós com os bits congelados(0) e bits de informação são mostrados em branco e cor preta, respectivamente. Os LLRs obtidos usando f e funções g da Eq. 2.32 e Eq. 2.35 são mostrados em azul e verde, respectivamente, enquanto os bits obtidos usando a parcial cálculos de soma da Eq. 2.36 e Eq. 2.37 são mostrados em vermelho. Os números que acompanham entre parênteses identificam a etapa de o processo de decodificação SC onde o LLR ou bit correspondente torna-se disponível.

Os decodificadores SC são preferidos por sua baixa complexidade de decodificação. No entanto, essa simplicidade é alcançada às custas de uma alta latência de decodificação, devido às várias dependências de dados presentes no processo de decodificação SC. Especificamente, as operações f precisam aguardar a disponibilidade dos valores LLR (Log-Likelihood Ratios) em suas conexões à direita, enquanto as operações g precisam esperar tanto pelos valores estimados de bits à esquerda quanto pelos LLRs à direita. Além disso, é necessário fornecer bits no lado esquerdo para facilitar a propagação de bits da esquerda para a direita. Por conta dessas dependências, os bits de informação no extremo esquerdo do circuito são recuperados de forma sequencial, de cima para baixo, o que torna a implementação de decodificadores SC em hardware desafiadora.

Mais especificamente, essas dependências de dados resultam em diferentes quantidades de operações que podem ser realizadas em paralelo em momentos distintos, como

Figura 19 – Árvore de decodificação SC



Fonte: (AL., 2019)

exemplificado na Figura 15. Para minimizar o número de etapas necessárias para completar o processo de decodificação, uma grande quantidade de hardware pode ser empregada, permitindo que uma única etapa de processamento execute o maior número de operações paralelas possíveis, suportadas pelas dependências de dados do decodificador. No entanto, essas dependências acabam limitando o uso eficiente de grande parte desse hardware durante o restante do processo de decodificação, o que pode justificar a escolha por uma quantidade menor de hardware e um maior número de etapas de processamento.

Independentemente da abordagem adotada, a relação entre o uso de recursos de hardware e a latência necessária para concluir o processo de decodificação pode ser desfavorável, a menos que sejam desenvolvidas técnicas alternativas e sofisticadas para otimizar esse balanço.

As principais características de um decodificador polar SC estão resumidas na Tabela.2.

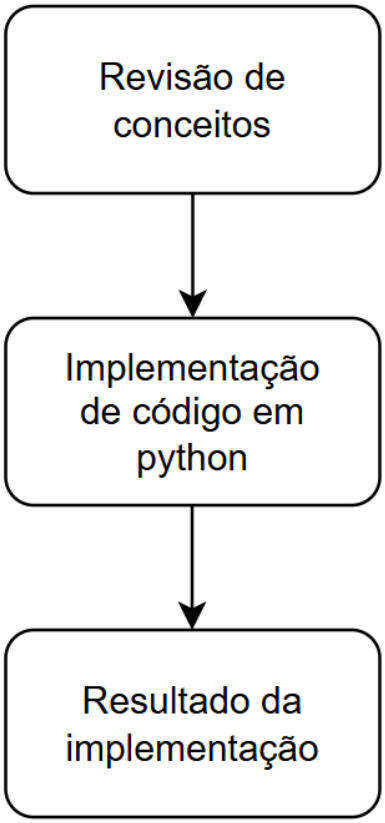
Tabela 2 – Principais características de um decodificador polar SC

Complexidade	• Complexidade de tempo = O(N log2 N) • Complexidade do espaço = O(N)
Vantagens	• Baixa complexidade de decodificação • Alcance de capacidade assintoticamente
Desvantagem	• Desempenho de comprimento finito abaixo do ideal • Processamento serial, resultando em alta latência (ou baixa taxa de transferência) • Implementação totalmente paralela inviável

3 Materiais e métodos

A Figura 20, apresenta o esboço do funcionamento geral da aplicação proposta. As atividades foram divididas em três etapas principais: Revisão de conceitos, que foram vistos na fundamentação teórica deste trabalho, a implementação do algoritmo para se trabalhar com códigos polares e os resultados obtidos através das simulações.

Figura 20 – Diagrama de funcionamento da proposta.



Fonte: Elaborado pelo autor

3.1 Implementação do Algoritmo de Códigos Polares

Neste projeto, foi utilizada a biblioteca *The project A Library for Polar Coding and Shortening Algorithms in Python* (MCBAIN., 2019), que oferece uma implementação de algoritmos em Python para pesquisadores e estudantes interessados em trabalhar com códigos polares, servindo como uma alternativa ao Matlab. Essa biblioteca é baseada nos algoritmos propostos por Vangala (VANGALA; VITERBO., 2015) e foi fundamental para as simulações do sistema de comunicação, permitindo a transmissão de diferentes tamanhos

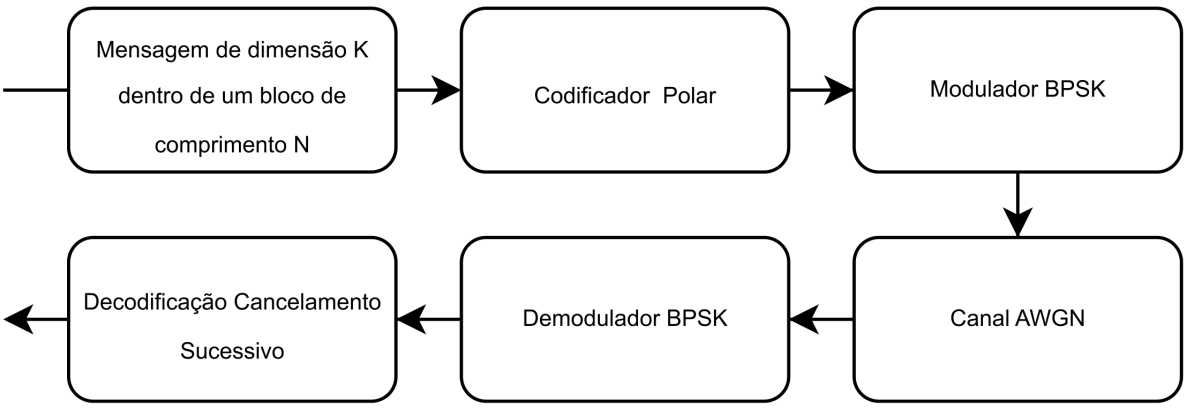
de palavras-código e a plotagem de curvas em função da Taxa de Erro de Quadro (FER) e da Taxa de Erro de Bit (BER).

As simulações foram realizadas no Jupyter Notebook, uma ferramenta versátil amplamente utilizada em projetos de ciência de dados e programação. No entanto, para atender às demandas específicas deste projeto, foi necessário realizar melhorias na biblioteca usada, principalmente na visualização dos resultados. Implementações adicionais foram feitas para aprimorar a geração das imagens referentes à Taxa de Erro de Bit (BER) e à Taxa de Erro de Quadro (FER), garantindo uma representação gráfica mais clara e precisa dos dados obtidos nas simulações.

Nos anexos em Capítulo 6, é possível visualizar todas as funções que foram aprimoradas, além da função principal responsável pela simulação individual de cada palavra-código. Ao final de cada simulação, os resultados são salvos localmente no computador, permitindo o acesso aos dados gerados pelas funções de cálculo de BER e FER para futuras análises e validações.

As palavras-código seguiram o fluxo esquematizado na Figura 21, sendo primeiramente codificadas, moduladas por um sistema BPSK e transmitidas por um canal AWGN. Finalmente, chegaram ao decodificador de Cancelamento Sucessivo. Este decodificador é o primeiro a ser utilizado para códigos polares, baseando-se na decodificação sequencial de cada bit e no uso de bits já decodificados para aprimorar as estimativas dos bits restantes. Especificamente, o bit decodificado na ramificação superior de cada iteração deve ser conhecido para que a decodificação do bit presente na ramificação inferior possa ser realizada com sucesso.

Figura 21 – Esquematização da Simulação.



Fonte: Elaborado pelo autor

Foram realizadas simulações para medir a confiabilidade e eficiência de um canal conforme o tamanho da palavra código N é modificada. Comumente um codificador transforma uma mensagem de tamanho K em um bloco de dimensão N ao adicionar bits de paridade. Contudo, na codificação polar são adicionados os $N - K$ bits congelados aos

bits de mensagem K antes da sua codificação.

4 Resultados

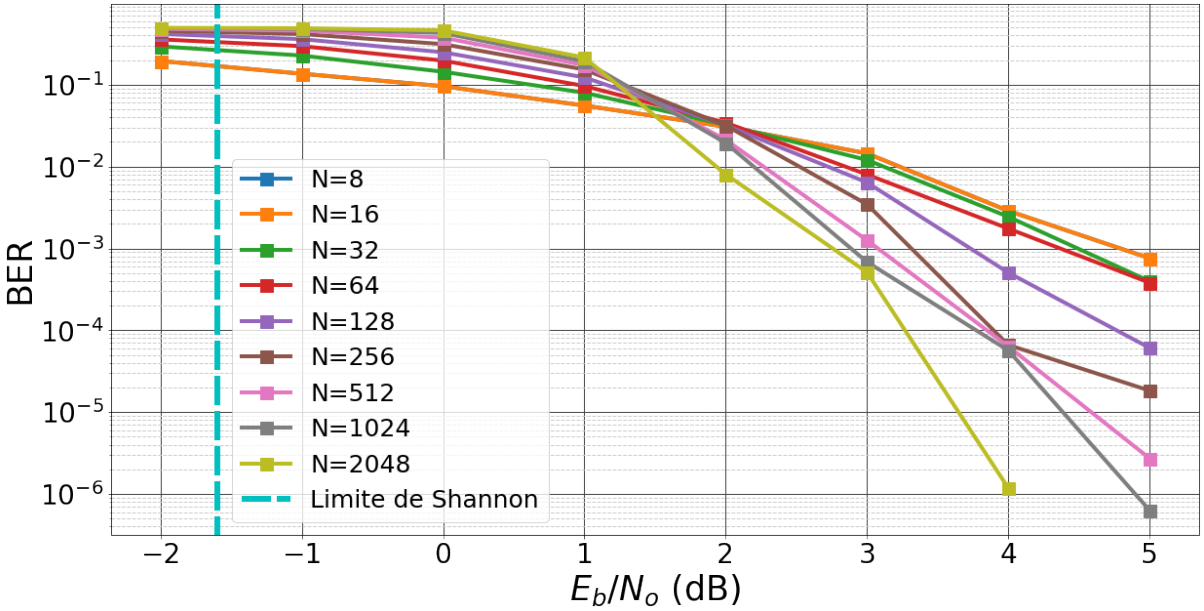
Nessa subseção serão apresentados os resultados finais obtidos através das simulações seção 3.1.

4.0.1 Resultados da Simulação

Na etapa de transmissão os bits passam por um modulador BPSK e em seguida por um canal onde receberá um ruído AWGN. A última etapa é passar pelo decodificador de cancelamento sucessivo para que a palavra código seja decodificada.

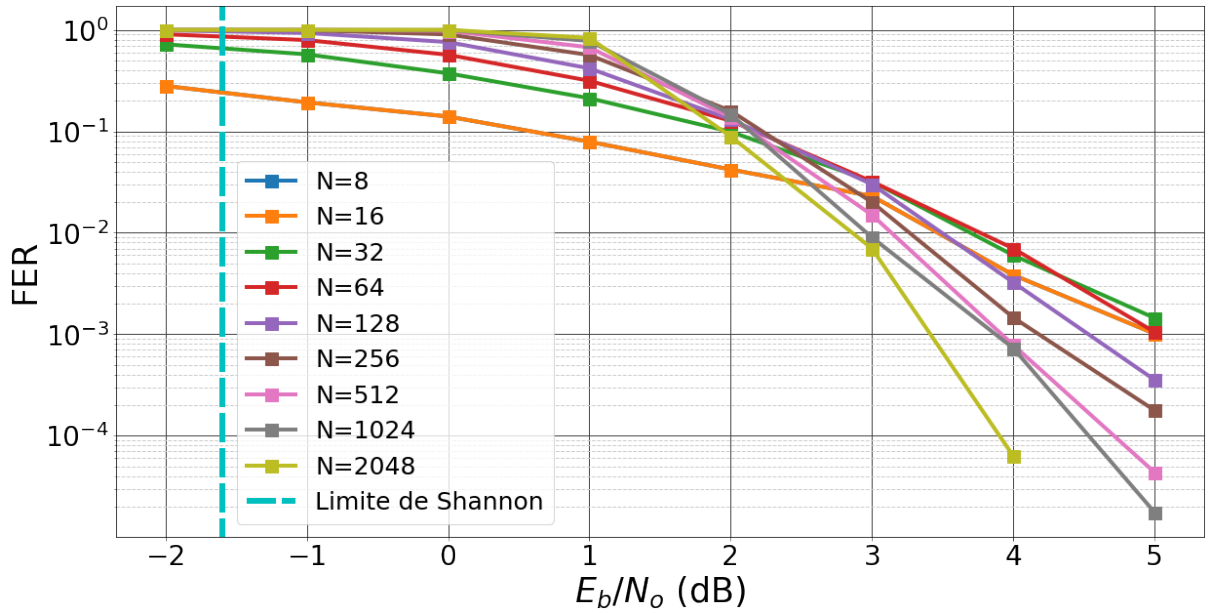
Nota-se na Figura 22 que ao aumentar o valor do tamanho N da palavra código no sentido crescente, as curvas se tornam mais íngremes, diminuindo a relação sinal/ruído, o FER (Frame Error Rate) também é atenuado para valores cada vez menores Fig. 23, deixando claro que o sinal é transmitido por uma conexão mais efetiva. A FER está diretamente ligada ao BER, uma vez que o bit é a menor unidade do quadro, conforme a BER é diminuída, a FER também é atenuada diretamente. Assim, conforme o tamanho de N varia crescentemente, o canal ficará mais confiável e eficiente.

Figura 22 – Resultados obtidos da taxa de erro de bit.



Fonte: Elaborado pelo autor

Figura 23 – Resultados obtidos da taxa de erro de quadro.



Fonte: Elaborado pelo autor

Através da Tabela 3, é visto como o BER e o FER se relacionam diretamente:

Tabela 3 – Resultados da BER e FER

Simulação de uma palavra código de tamanho N=8					
Eb/No	FER	BER	Frame Errors	Bits Errors	Iterações
-2	0.28	0.19525	280	781	1000
-1	0.28	0.19525	280	781	1000
0	0.193	0.1365	193	546	1000
1	0.14	0.096 0	140	384	1000
2	0.079	0.05575	79	223	1000
3	0.042	0.03075	42	123	1000
4	0.023	0.0145	23	58	1000
5	0.004	0.00288	5	15	1303
6	0.001	0.00076	5	15	4945

A polarização do canal utilizada na codificação permite entender porque para valores maiores de N o canal fica mais confiável. De acordo com a teoria demonstrada na fundamentação teorica, para valores de N tendendo ao infinito os canais polarizados tenderão para os extremos correspondentes a 0 ou 1, assim para maiores valores de N haverá um aumento na confiabilidade do canal, dado que os canais ruins utilizados para enviar os bits congelados serão bem distintos dos canais bons utilizados para enviar os bits de informação.

5 Considerações Finais

O presente trabalho teve como objetivo estudar os processos de codificação e decodificação de códigos polares e apresentá-los de forma didática, visando à sua aplicação em sistemas de comunicação. Para tal, no tópico de fundamentação teórica foram descritos conceitos básicos referentes a teoria da informação, demonstrando os significados de informação, entropia, informação mútua e capacidade de canal. Em seguida foram descritos o funcionamento do código de blocos lineares que servem como base para o entendimento dos códigos polares já que o código polar em si é um tipo de código de bloco linear. Posteriormente a polarização de canal demonstra a separação dos canais bons e dos canais ruins, mostrando que os caminhos seguidos pelos bits de informação são diferentes dos caminhos seguidos pelos bits congelados, e que conforme o tamanho da palavra código N aumenta, mais distintos se tornam os caminhos.

É importante ressaltar que os códigos polares são projetados para corrigir erros de bits que ocorrem durante a transmissão de dados, aumentando a confiabilidade das informações transmitidas. Eles utilizam técnicas de polarização de canal para identificar os melhores canais para a transmissão dos bits de informação. Embora os códigos polares sejam altamente eficientes, em alguns sistemas práticos, eles podem ser combinados com outros códigos corretores de erros para proporcionar uma camada adicional de segurança e melhorar ainda mais o desempenho global da comunicação.

Com a aplicação dos conceitos teóricos e o auxílio da biblioteca (MCBAIN., 2019) foi possível demonstrar através das simulações que canais de tamanho N = 2048, N = 1024 e N = 512 se provaram os mais eficientes em ambientes onde a relação Eo/No é maior que $4dB$, convergindo mais rápido para um valor de FER de 10^{-5} e BER de 10^{-6}, assim conforme o tamanho das palavras código a serem transmitidas é elevado, menor é o valor de Eb/No necessário para atingir a mesma BER de palavras menores, provando assim um ganho energético na transmissão de palavras maiores.

O estudo dos códigos polares desenvolvido neste trabalho foi a base para a elaboração de um artigo, publicado na Sociedade Brasileira de Telecomunicações (SBrT) em 2023, intitulado "Eficiência da Polarização de Canal nos Códigos Polares Sistemáticos e Decodificação por Cancelamento Sucessivo para Redes 5G"(SOUZA LUIZ RODRIGUES, 2023). O artigo destaca a importância desses códigos no contexto das redes 5G e reforça a relevância da pesquisa no avanço das telecomunicações.

Referências

AL., B. Z. et. *Polar codes and their quantum-domain counterparts.* 2019. Disponível em: <https://ieeexplore.ieee.org/document/8815775>. Acesso em: maio 2024. Citado 8 vezes nas páginas 15, 30, 33, 34, 35, 36, 37 e 42.

AL., N. K. et. *Polar codes: Primary concepts and practical decoding algorithms.* 2014. Disponível em: <https://ieeexplore.ieee.org/document/6852102>. Acesso em: Junho 2024. Citado na página 15.

ARIKAN., E. *A performance comparison of polar codes and reed-muller codes.* 2008. Disponível em: <https://ieeexplore.ieee.org/document/4542778>. Acesso em: maio 2024. Citado na página 34.

ARIKAN, E. *Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels.* 2009. Disponível em: <https://arxiv.org/pdf/0807.3917>. Acesso em: 3 maio 2023. Citado 2 vezes nas páginas 30 e 35.

ARIKAN., E. *ystematic polar coding.* 2011. Disponível em: <https://ieeexplore.ieee.org/document/5934670>. Acesso em: maio 2024. Citado na página 32.

CHEN, Z. Z. C. Z. G. T.; ZHANG., L. *A low complexity encoding algorithm for systematic polar codes.* 2016. Disponível em: <https://ieeexplore.ieee.org/document/7462203>. Acesso em: maio 2024. Citado na página 32.

FARIAS, R. B. de. *Análise da necessidade de mudança regulatória para a implantação do 5G no Brasil.* 2021. Disponível em: <https://bibliotecadigital.fgv.br/dspace;handle/handle/10438/30679>. Acesso em: 3 maio 2023. Citado na página 13.

HAGENAUER, E. O. J.; PAPKE., L. *Iterative decoding of binary block and convolutional codes.* 1996. Disponível em: <https://ieeexplore.ieee.org/document/485714>. Acesso em: maio 2024. Citado na página 39.

HOLMA, N. T. N. H. *5G technology : 3GPP new radio.* 1. ed. Japan: John Wiley Sons Ltd, 2020. 517 p. Citado na página 13.

LEONARDO. erças. *odificação de Canal para Comunicações Ultra Confiáveis em Sistemas 5G.* 2019. Disponível em: <https://proceedings.sbmac.org.br/sbmac/article/view/3609>. Acesso em: 7 maio 2024. Citado na página 15.

LIN, W. R. e S. *HANNEL CODES.* 1. ed. [S.l.]: Cambridge University Press, 2001. Citado na página 15.

MARTãO, V. B. *Codificação de canal para redes 5G utilizando códigos polares.* 2018. Disponível em: <https://repositorio.unesp.br/items/91186287-ff7c-4609-8e24-16d0e8b0db7a>. Acesso em: 5 maio 2023. Citado 2 vezes nas páginas 13 e 15.

MATTOS, L. M. T. da S. *Códigos polares para sistemas de comunicações sem fio do futuro.* 2018. Disponível em: <https://www.puc-rio.br/ensinopesq/ccpg/pibic/relatorio_resumo2018/rel_ctc_cetuc.html>. Acesso em: 5 maio 2023. Citado na página 13.

MCBAIN., B. *olar codes in python: Main reference*. 2019. Disponível em: <https://github.com/mcba1n/polar-codes/tree/master>. Acesso em: maio 2024. Citado 2 vezes nas páginas 44 e 50.

SARKIS, I. T. G.; GIARD., P. *Flexible and low-complexity encoding and decoding of systematic polar codes*. 2016. Disponível em: <https://arxiv.org/abs/1507.03614>. Acesso em: maio 2024. Citado na página 32.

S.HAYKIN. *COMMUNICATION SYSTEMS*. 1. ed. [S.l.]: Wiley, 2001. Citado na página 15.

SOUZA LUIZ RODRIGUES, M. A. S. C. T. S. A. *Eficiência da Polarização de Canal nos Códigos Polares Sistemáticos e Decodificação por Cancelamento Sucessivo para Redes 5G*. 2023. Disponível em: <https://biblioteca.sbrt.org.br/articlefile/4548.pdf>. Acesso em: maio 2024. Citado na página 50.

TACIANA, S. *ÁLGEBRA DE CORPOS FINITOS APLICADA À TEORIA DA CODIFICAÇÃO: ESTUDO DO CODIFICADOR BCH*. 2012. Disponível em: <https://repositorio.ufpb.br/jspui/handle/123456789/414?locale=pt_BR>. Acesso em: maio 2024. Citado na página 15.

TERÇAS., B. *Codificação de Canal para Comunicações Ultra Confiáveis em Sistemas 5G*. 2021. Disponível em: <https://proceedings.sbmac.org.br/sbmac/article/view/3609>. Acesso em: maio 2024. Citado 3 vezes nas páginas 27, 28 e 29.

THOMAS, T. e J. *LEMENTS OF INFORMATION THEORY*. 1. ed. [S.l.]: Wiley-Interscience, 2006. Citado na página 15.

VANGALA, Y. H. H.; VITERBO., E. *Efficient algorithms for systematic polar encoding*. 2015. Disponível em: <https://ieeexplore.ieee.org/document/7315017>. Acesso em: maio 2024. Citado 2 vezes nas páginas 32 e 44.

6 Anexos

Código 1 – Função para plotar FER

```
def plot_helper( new_plot, sim_filenames, dir, plot_title = ''):
    # plot the FER and BER from file list
    new_plot.cla()
    for sim_filename in sim_filenames:
        with open(dir + sim_filename + '.json') as data_file:
            data_loaded = json.load(data_file)
        new_plot.plot(data_loaded['SNR'], data_loaded['FER'], '-s',
            markersize=13, linewidth=4, label=sim_filename)
    # format the plots
    new_plot.set_title(plot_title,fontsize="35")
    plt.yticks(fontsize=28)
    plt.xticks(fontsize=28)
    new_plot.set_ylabel("FER",fontsize="35")
    new_plot.set_xlabel("$E_b/N_o$ (dB)",fontsize="35")
    new_plot.grid(b=True, which='major', color='#444',linestyle='-',
        linewidth=1)
    new_plot.grid(b=True, which='minor', color='#ccc',linestyle='--',
        linewidth=1)
    new_plot.set_yscale('log')
    new_plot.axvline(-1.6, ymin=-1, ymax=1.5, linestyle= (0, (5, 1)),
        linewidth=6,color='c',label = "Limite de Shannon")
    new_plot.legend(loc='lower left',fontsize="25",bbox_to_anchor=(0.1,0))
    plt.show()

def plot1( sim_filenames, dir):
    """
    Plot multiple sets of FER data from the same directory on the same axes
        .
    Parameters
    ----------
    sim_filenames: ndarray<string>
        a list of all filenames to plot in a common root directory
    dir: string
        the root directory for the specified filenames
    """
    fig= plt.figure(figsize=(20, 10) )
    new_plot = fig.add_subplot(111)
    plot_helper(new_plot, sim_filenames, dir)
    fig.show()
plot1( ['N=8','N=16','N=32','N=64','N=128','N=256','N=512','N=1024','N=2048
    '],'data/' )
```

Código 2 – Função para plotar BER

```
def plot_helper( new_plot, sim_filenames, dir, plot_title = ''):
    # plot the FER and BER from file list
    new_plot.cla()
    for sim_filename in sim_filenames:
        with open(dir + sim_filename + '.json') as data_file:
            data_loaded = json.load(data_file)
        new_plot.plot(data_loaded['SNR'], data_loaded['BER'], '-s',
            markersize=13, linewidth=4, label=sim_filename)

    # format the plots
    new_plot.set_title(plot_title,fontsize="35")
    plt.yticks(fontsize=28)
    plt.xticks(fontsize=28)
    new_plot.set_ylabel("BER",fontsize="35")
    new_plot.set_xlabel("$E_b/N_o$ (dB)",fontsize="35")
    new_plot.grid(b=True, which='major', color='#444',linestyle='-',
        linewidth=1)
    new_plot.grid(b=True, which='minor', color='#ccc',linestyle='--',
        linewidth=1)
    new_plot.set_yscale('log')
    new_plot.axvline(-1.6, ymin=-1, ymax=1.5, linestyle= (0, (5, 1)),
        linewidth=6,color='c',label = "Limite de Shannon")
    new_plot.legend(loc='lower left',fontsize="25",bbox_to_anchor=(0.1,0))
    plt.show()

def plot1( sim_filenames, dir):
    """
    Plot multiple sets of BER data from the same directory on the same axes
        .
    Parameters
    ----------
    sim_filenames: ndarray<string>
        a list of all filenames to plot in a common root directory
    dir: string
        the root directory for the specified filenames

    """
    fig= plt.figure(figsize=(20, 10) )
    new_plot = fig.add_subplot(111)
    plot_helper(new_plot, sim_filenames, dir)
    fig.show()
plot1( ['N=8','N=16','N=32','N=64','N=128','N=256','N=512','N=1024','N=2048
    '],'data/' )
```

Código 3 – Simulação principal com exemplo para N=8 k=4

```
import numpy as np
from polarcodes import *
import matplotlib.pyplot as plt
import json

# inicializa o c digo polar
myPC = PolarCode(8,4)
myPC.construction_type = 'bb'

# constru  o de c digo-m e
design_SNR  = 1
Construct(myPC, design_SNR)
print(myPC, "\n\n")

# definir mensagem
my_message = [1,0,0,1]
myPC.set_message(my_message)
print("A mensagem  :", my_message)

# codificar mensagem
Encode(myPC)
print("A mensagem codificada  :", myPC.get_codeword())

# transmitir a palavra c digo
AWGN(myPC, design_SNR)
print("As probabilidades de log s  o:", myPC.likelihoods)

# decodifica a palavra de c digo recebida
Decode(myPC)
print("A mensagem decodificada  :", myPC.message_received)

# simula  o do c digo polar
myPC.simulate(save_to='data/N=8', Eb_No_vec=np.arange(-2,6),
    manual_const_flag=True, max_iter=100000000,min_iterations=1000,
    min_errors=5)
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus João Pessoa - Código INEP: 25096850
Av. Primeiro de Maio, 720, Jaguaribe, CEP 58015-435, Joao Pessoa (PB)
CNPJ: 10.783.898/0002-56 - Telefone: (83) 3612.1200

Documento Digitalizado Ostensivo (Público)

tcc versão final com ficha catalográfica

Assunto:	tcc versão final com ficha catalográfica
Assinado por:	Alysson Batista
Tipo do Documento:	Anexo
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Alysson Batista de Souza, ALUNO (20142610296) DE BACHARELADO EM ENGENHARIA ELÉTRICA - JOÃO PESSOA**, em 20/11/2024 13:20:58.

Este documento foi armazenado no SUAP em 20/11/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1314729
Código de Autenticação: 95484fb73e